高校生ものづくりコンテスト

「電子回路組立」部門

第１１回東京大会

指　導　書

ZENJYOUKEN

平成２４年３月

全国情報技術教育研究会

ご　挨　拶

埼玉県立熊谷工業高等学校長　金子　正明
（全国情報技術教育研究会長）

全国情報技術教育研究会では、（社）全国工業高等学校長協会主催の高校生ものづくりコンテスト「電子回路組立」部門を、第５回東京大会から支援させて頂いています。この大会から部門の課題が、コンピュータのソフトウェア技術とハードウェア技術、そして両者を結ぶインターフェース技術の知識と技術を必要とする「組込み技術」に関する内容となりました。全国情報技術教育研究会では、競技内容が情報技術関連科目に関することから、できる支援として、ホームページ上において各県・地区大会の様子や外部制御基板の頒布紹介などをしてまいりました。会員の皆様の御協力により年を追う事に大会も活発になってきており、併せて「組込み技術」に関する指導資料を求める声も出てまいりました。

これを受け全国情報技術教育研究会では、昨年度・一昨年度と関係各位の御協力によりまして、高校生ものづくりコンテスト「電子回路組立」部門の課題についての指導書を作成いたしました。今年度も同様に第１１回東京大会の課題について、主にプログラムと関連事項を解説した内容を(株)ルネサスソリューションズ　ルネサスマイコンカーラリー事務局様の御協力を得まして、本書にまとめることができました。

本書は、次年度以降に開催される各地区・県大会の指導資料として、そして何よりも、先生方の「組込み技術」教育の研修や実験・実習等の授業の指導資料として活用いただければ幸いです。

本書を作成するにあたり、御協力いただきました皆様に深く感謝申し上げます。

平成２４ 年３月吉日

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　課題

１．課題

競技時間中に製作する『設計製作回路①』と大会当日に配布する『制御対象回路③』を、事前に製作したケーブルにより『持参コンピュータ②』と接続し、競技時間内に『制御プログラム④』を作成し、目的の動作を行うシステムを完成させる。

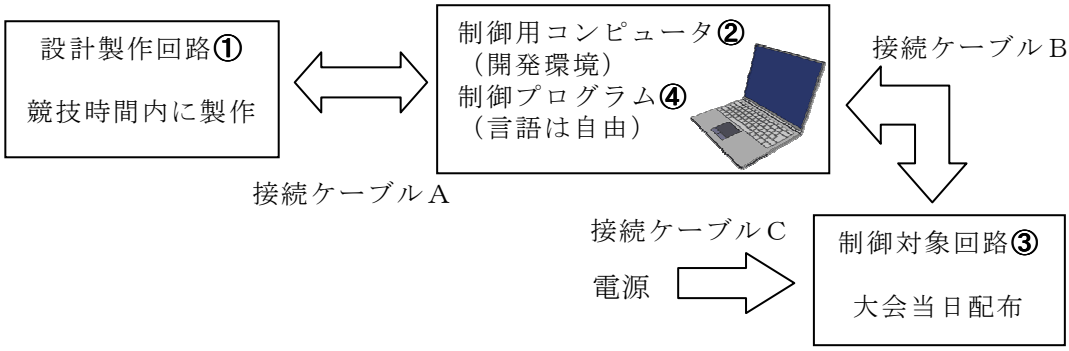


図１　課題概略図

（１）設計製作回路①

大会当日に示す設計仕様に基づく電子回路を設計し、ユニバーサル基板を用いて電子回路基板を製作する。配線はスズメッキ線を使用し、設計製作回路は以下の部品を使用する。

①ユニバーサル基板（サンハヤト ICB293 相当品）

②フォトインタラプタ、スイッチ、コネクタ、0.5φスズメッキ線等

（２）制御用コンピュータ②

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。

（３）制御対象回路③

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

（４）制御プログラム④

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。

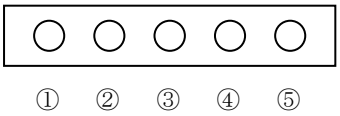
①ストップウォッチのプログラム

②回転制御のプログラム

（５）接続ケーブル

①接続ケーブルＡ（設計製作回路❶用）

ＩＣピッチ１列５ピンコネクタ・メス（ストレートピンヘッダハウジング）

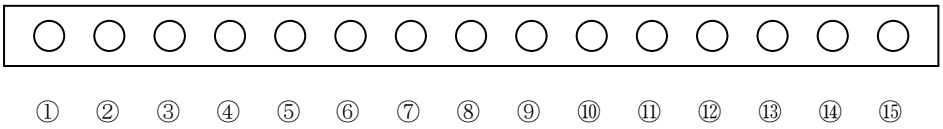


①	ＧＮＤ	②	５Ｖ	③	入力０
④	入力１	⑤	入力２		

図２　接続ケーブルＡ用コネクタのピン配置

②接続ケーブルＢ（制御対象回路❸用）

ＩＣピッチ１列１５ピンコネクタ・メス（ストレートピンヘッダハウジング）

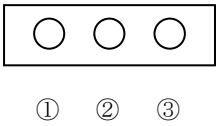


①	ＧＮＤ	②	出力０	③	出力１	④	出力２	⑤	出力３
⑥	出力４	⑦	出力５	⑧	出力６	⑨	出力７	⑩	出力８
⑪	出力９	⑫	ＮＣ	⑬	ＮＣ	⑭	ＮＣ	⑮	ＮＣ

図３　接続ケーブルＢ用コネクタのピン配置

③接続ケーブルＣ（電源供給用）

ＩＣピッチ１列３ピンコネクタ・メス（ストレートピンヘッダハウジング）



①	ＧＮＤ	②	５Ｖ	③	ＧＮＤ

図４　接続ケーブルＣ用コネクタのピン配置

注）コネクタ・メスの具体的なイメージはヒロセ電機のホームページを参照。

(http://www.hirose.co.jp/catalogj_hp/j21800074.pdf)

写真は HNC2－2.5P－8DS

（ＩＣピッチ１列８ピンコネクタ・メス）

２．作業条件

（１）競技時間　2 時間 30 分（150 分）

（２）競技実行委員から配布するもの

・『設計製作回路①』で使用する部品および材料等
・『制御対象回路③』
・コンテストで使用する部品の規格表
・Ａ４サイズ方眼紙
・AC100V コンセント(2 口)

（３）競技者が準備するもの

・『制御用コンピュータ②』および開発環境
・接続ケーブルＡ、接続ケーブルＢ、接続ケーブルＣ
・＋５Ｖの電源（Max1000mA 程度）
・工具類およびテーブルタップ
・ソースリスト提出用の記録媒体（USB メモリまたは FD）
・筆記用具および定規・テンプレート類

工具類とは、各自の作業に必要なもので、はんだごて、こて台、はんだ吸い取り器、ニッパ、リードペンチ、＋ドライバ、テスタ、保護メガネ、基板支持台　等

（４）競技者の服装等

・競技中は、各学校で使用している作業服を着用する。
・はんだ付けの作業時には、保護メガネを着用する。ただし、メガネをかけている場合はこの限りではない。

（５）注意事項

①作業を行うにあたっては、安全に十分注意する。
②配布された部品および材料以外のものは、使用しない。
③規格表・命令表が必要な場合は各自で準備し、大会前日に承認を受ける。
④事前に製作したヘッダーファイルは、大会前日に申請し内容の承認を受ける。
⑤接続ケーブルＡ・Ｂ・Ｃは、図２・３・４を参考に事前に製作し準備しておく。ただし、ケーブルの長さは自由である。
⑥ソースリストは、テキスト形式で記録媒体に保存・提出する。

３．審査対象

（１）『設計製作回路①』の設計図

（２）『設計製作回路①』の製作基板

（３）仕様に対応する動作

（４）プログラムのソースリスト

（５）その他（作業態度等）

４．採点基準

（１）採点項目と観点

項　　目	配点	観　　点
プログラミング技術	40	・動作 ・構造 ・書式 ・読みやすさ
組み立て技術	30	・工具類の使い方 ・部品処理 ・はんだの状態 ・配線 ・配置
設計力	20	・正確さ ・配置 ・記号 ・文字
その他	10	・作業態度 ・作業工程
合　計	100	

（２）順位の決定方法

①合計得点の高い順に、１位、２位、３位･･･とする。

②同点の場合は、「プログラミング技術」の得点の高い選手を高位とする。

③「プログラミング技術」の得点も同点の場合は、「組み立て技術」の得点の高い選手を高位とする。

④さらに同点の場合は、「設計力」の得点の高い選手を高位とする。それでもなお同点の場合は、全体の完成度から順位を決定する。

（３）その他

設計製作回路①の製作に関して、別紙『**標準的なはんだ付けについて**』を参照。

５．その他

（１）鉛フリーはんだについて

無鉛（鉛フリー）はんだ（Sn-3.0Ag-0.5Cu、0.8mm）を使用する。

（２）動作確認について

プレ審査時に競技実行委員の指示に従い､競技者が操作して課題の動作確認を行う。

（３）設計製作回路・制御対象回路・当日の課題プログラムについて

『設計製作回路①』･『制御対象回路③』の回路図・使用部品の規格等については、事前公開しない。

また、当日作成する制御プログラムに関しても、事前公開はしない。

（４）その他

大会の参考として、全国情報技術教育研究会ホームページを参照。

標準的なはんだ付けについて

（１）はんだのぬれ性

イ．はんだが接合するリード線、銅箔によく流れ、長くすそを引いていること。

ロ．部品穴のはんだ付けは、ランドの表面にはんだのぬれ性があること。

（２）はんだの量

イ．はんだの量は、部品リード線の折り曲げ部分、線の切り口等をはんだで覆い、かつ、肉厚が薄く線の形がわかるものとする。（図を参照）
ただし、折り曲げず、かつ、切断せずに取り付ける部品リードのはんだ付けを行う場合は、リードの先端まで、全面はんだで覆わなくてもよい。

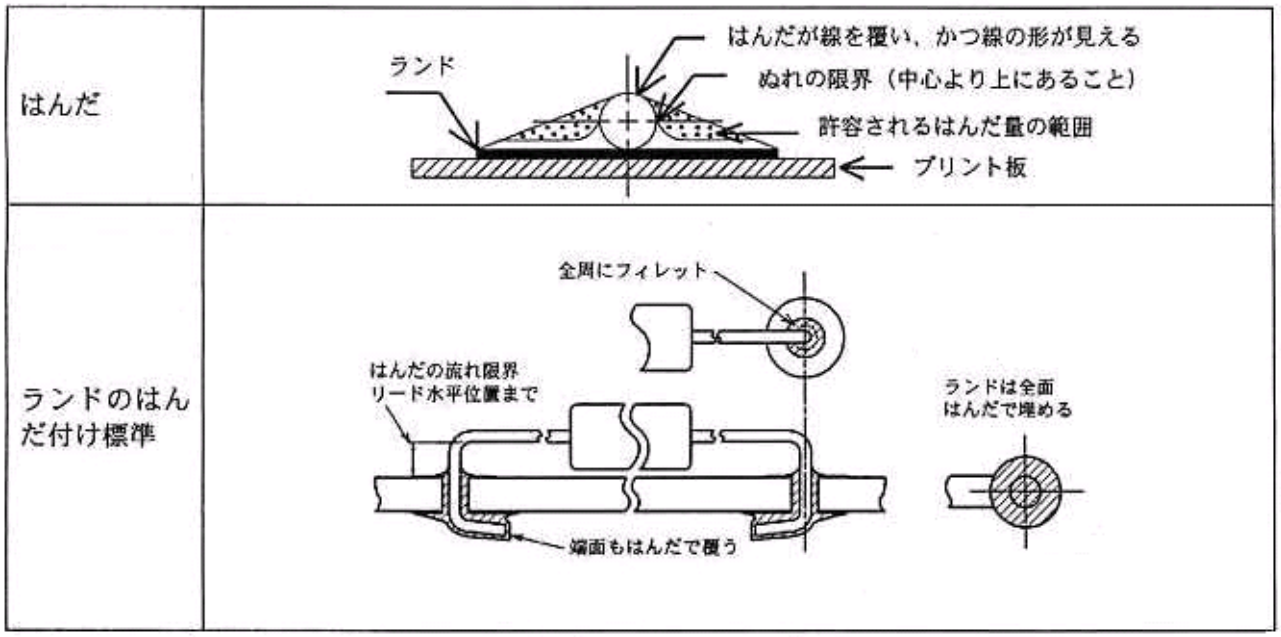
（３）その他

イ．部品端子の線材接合部は、穴あきのないようにはんだ付けすること。

ロ．ランドのないところで、線又は部品リードを接続しないこと。
（空中配線接続をしてはならない）

ハ．ランドをはく離させないこと。

ニ．できるだけジャンパー線を用いず、裏面のみで配線をおこなうこと。



平成 23 年度　高校生ものづくりコンテスト

電子回路組立部門　課題

1. システム機構

競技時間中に製作する『設計製作回路①』と大会当日に配布する『制御対象回路③』を、事前に製作したケーブルにより『持参コンピュータ②』と接続し、競技時間内に『制御プログラム④』を作成し、目的の動作を行うシステムを完成させる。

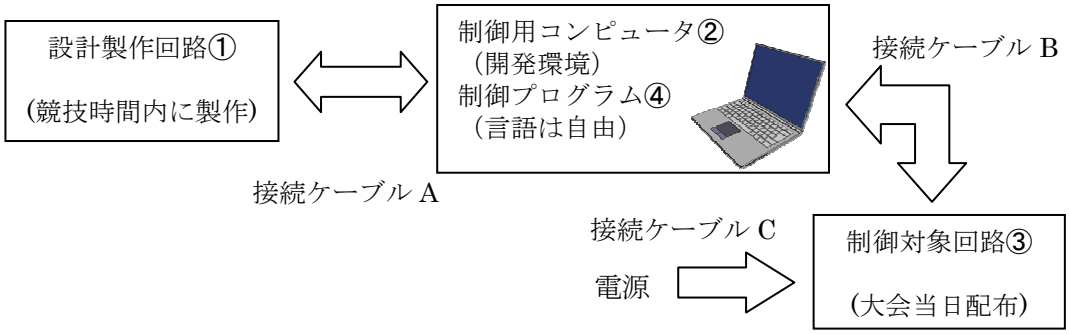


図 1　課題概略図

2. 設計・製作する入力回路

以下の(1)から(4)の条件を満たした回路を設計し、完成させなさい。

(1) 下図に指示したピン配置による入力回路を設計・製作する。

(2) 支給した方眼紙(A4 横)に回路図を描き提出する。

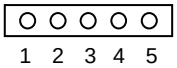
(3) 支給された部品を使って設計した回路を製作する。

(4) **基板の左下に赤丸シールを張り、基板原点とする。**

(ア) トグルスイッチは、つまみが上下に可動する。

(イ) フォトインタラプタは、ユニバーサル基板上の部品面に直接配線する。

2.54 inch　1 列 5 ピンストレート



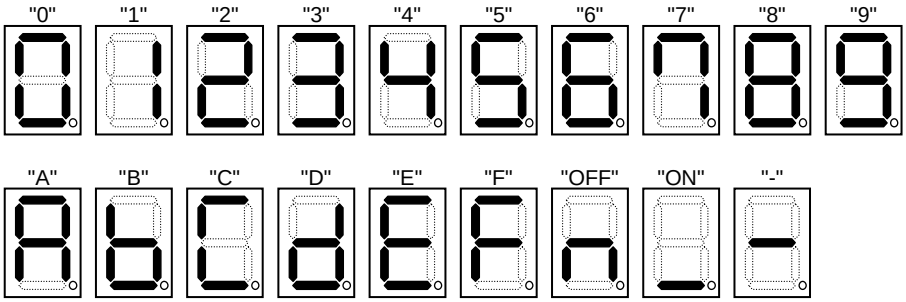
ピン番号	端子名	略称
1	GND	GND
2	+5V	5V
3	透過型フォトインタラプタ	PHSW
4	タクトスイッチ	TCSW
5	トグルスイッチ	TGSW

設計・製作する回路のコネクタピン配置図(基板面)

3. 作成するプログラム

(1) 課題 1 から課題 7 のプログラムを作成しなさい。また、課題は順番通りに作成しなくても良い。

(2) 7 セグメント LED に表示する数値およびアルファベットなどは以下のとおりとする。



(3) ステッピングモータおよび DC モータの回転は、目視および触って確認できること。

(4) 各課題において、トグルスイッチおよびタクトスイッチ、フォトインタラプタの状態を表す文中の表現は、以下の通りとする。

(ア) トグルスイッチ(TGSW)

① ON ･･･ トグルスイッチのつまみが上

② OFF ･･･ トグルスイッチのつまみが下

(イ) タクトスイッチ(TCSW)

① ON ･･･ タクトスイッチを押す

② OFF ･･･ タクトスイッチを離す

③ ON→OFF ･･･ タクトスイッチを 1 回押して離す

(ウ) フォトインタラプタ(PHSW)

① 透過 ･･･ フォトインタラプタの光が透過する

② 遮断 ･･･ フォトインタラプタの光が遮断する

③ 遮断→透過 ･･･ フォトインタラプタの光が 1 回遮断し、透過する

(5) 各課題において、初期状態は以下の通りとする。ただし、課題で直接指示がある場合はこの限りではない。

(ア) トグルスイッチ(TGSW) ･･･ OFF
(イ) タクトスイッチ(TCSW) ･･･ OFF
(ウ) フォトインタラプタ(PHSW) ･･･ 透過
(エ) 7 セグメント LED ･･･ 左右ともに消灯
(オ) ステッピングモータ(STM) ･･･ 停止
(カ) DC モータ(DCM) ･･･ 停止

(6) ステッピングモータに取り付けてある目盛盤は手で動かすことができる。(ステッピングモータの指示針は手で動かしてはいけない。)

(7) 各課題において、対象外のものは動作させない。

(8) 課題中に示されている動作概念図はおよその概要が示されている。条件の詳細については、課題文章のとおりに動作させる。

課題１　7 セグメント LED の表示

(1) タクトスイッチの「ON」、「OFF」の状態を 7 セグメント LED に表示する。

タクトスイッチが「OFF」　　タクトスイッチが「ON」

タクトスイッチの状態を示す 7 セグメント LED の表示

(2) 表示する 7 セグメント LED の位置は、フォトインタラプタの状態で次のように決まる。

(ア)　フォトインタラプタが「透過」で、右側の 7 セグメント LED に表示する。

(イ)　フォトインタラプタが「遮断」で、左側の 7 セグメント LED に表示する。

課題２　ステッピングモータの回転

(1) プログラムを開始した後、ステッピングモータの目盛盤を手で動かし 0 の位置を指示針に合わせる。

(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を開始し一回転して停止する。

(3) 回転方向は、タクトスイッチを「ON」する直前のトグルスイッチの状態により決まる。

(ア)　トグルスイッチが「OFF」で、時計回りに回転する。

(イ)　トグルスイッチが「ON」で、反時計回りに回転する。

(4) ステッピングモータが一回転し停止した後、(2)以降の動作が行える。

課題３　DC モータの回転

(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。

(ア)　フォトインタラプタが「透過」で、低速に回転する。

(イ)　フォトインタラプタが「遮断」で、高速に回転する。

(ウ)　上記(ア)と(イ)は回転中にいつでも変更ができる。

課題４　7 セグメント LED の二桁表示とカウントダウン

(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」で、10 進数で"10"から"00"までのカウントダウンを行う。

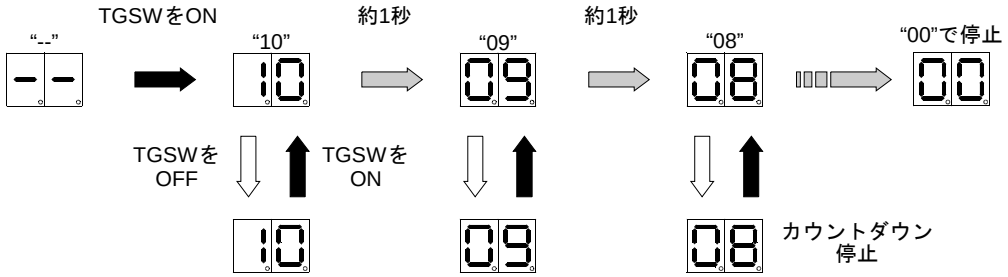
(ア)　カウントダウンの状態を左右の 7 セグメント LED に表示する。

(イ)　約 1 秒ごとにカウントダウンし、"00"で停止する。

(3) カウントダウン中にトグルスイッチが「OFF」で、カウントダウンを停止する。

(ア)　停止中の 7 セグメント LED の表示は、停止した直後の表示を維持する。

(イ)　停止中に再びトグルスイッチが「ON」で、そこからカウントダウンを引き続き行う。



課題５　ステッピングモータの位置制御　(1-2 相励磁)

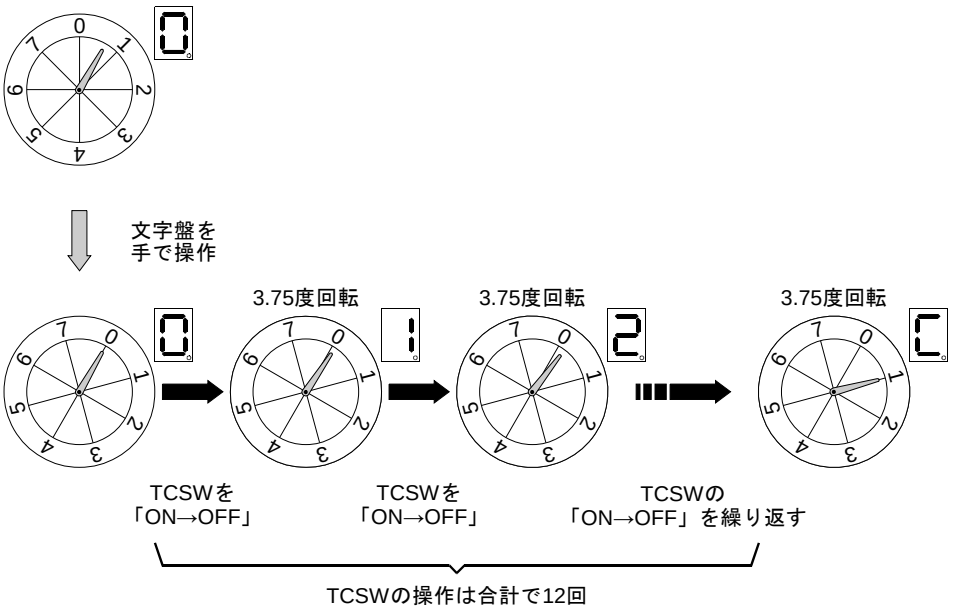
(1) プログラム開始直後、右側の 7 セグメント LED に"0"を表示する。その後、ステッピングモータの目盛盤を手で動かし 0 の位置を指示針に合わせる。

(2) タクトスイッチの「ON→OFF」を 12 回行い、ステッピングモータを時計回りに 45 度回転させる。

(ア) 1 回のタクトスイッチの「ON→OFF」で、ステッピングモータは 3.75 度回転する。

(イ) 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。

(ウ) 上記(ア)と(イ)は、タクトスイッチの「ON→OFF」の「OFF」の瞬間に動作する。



課題６　タクトスイッチによるモード選択

(1) タクトスイッチが「ON」の瞬間に回転しているモータがあれば、モータを停止し、「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。

(ア) 「ON」にしてから「OFF」になるまでの時間が約 1 秒未満なら、ステッピングモータも DC モータも回転しない。

(イ) 「ON」にしてから「OFF」になるまでの時間が約 1 秒以上 3 秒未満なら、ステッピングモータのみが反時計回りに回転する。

(ウ) 「ON」にしてから「OFF」になるまでの時間が約 3 秒以上なら、DC モータのみが反時計回りに回転する。

課題７　ステッピングモータの回転と復帰動作

(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目盛盤を手で動かし 0 の位置を指示針に合わせる。

(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。

(ア) 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。

(イ) 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。

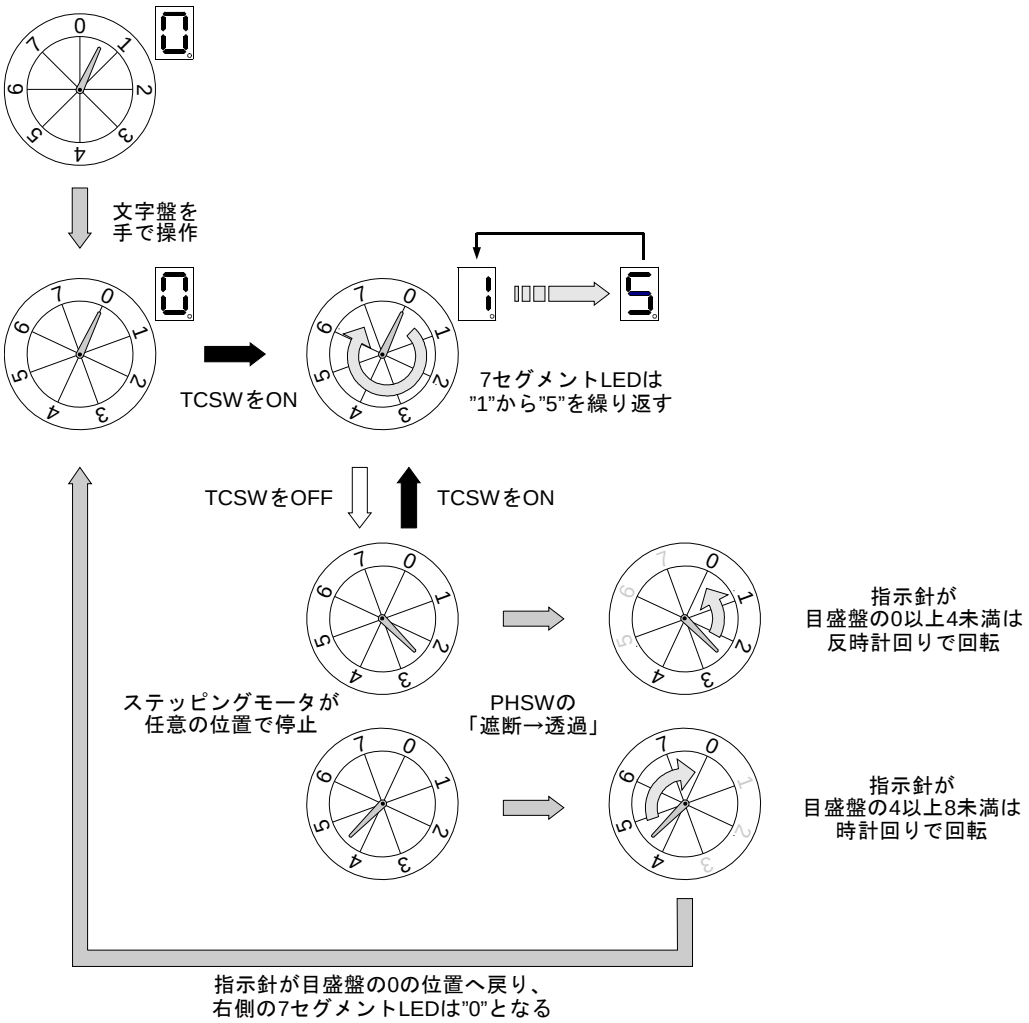
(ウ) 右側の 7 セグメント LED の表示は、"5"の次は"1"に戻り繰り返す。

(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」すると、**「遮断」の瞬間**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。

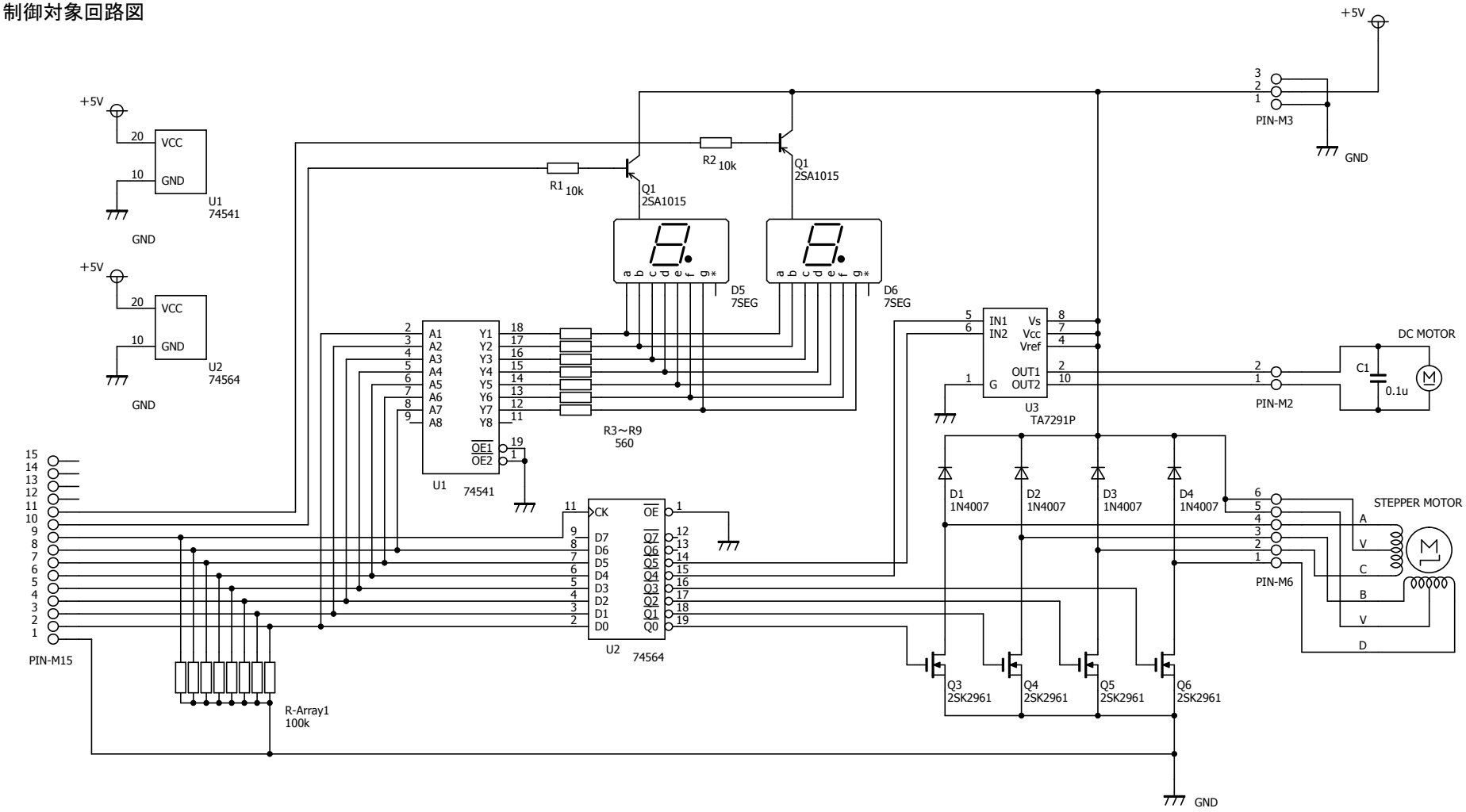
(ア) ステッピングモータの指示針が目盛盤の 0 以上 4 未満の場合は、反時計回りに回転する。

(イ) ステッピングモータの指示針が目盛盤の 4 以上 8(0 の位置)未満の場合は、時計回りに回転する。

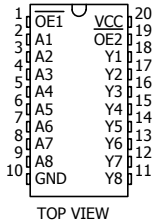
(5) 上記(4)の動作後、ステッピングモータの指示針が目盛盤の 0 の位置で戻ると、右側の 7 セグメント LED の表示は"0"となり、(2)以降の動作が行える。



平成23年度
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
制御対象回路図



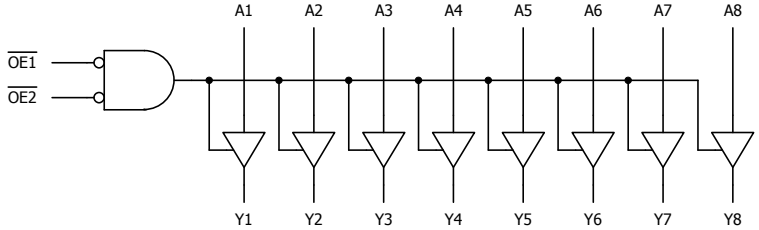
Octal Bus Buffer, Non-Inverting, 3-State Outputs
74HC541



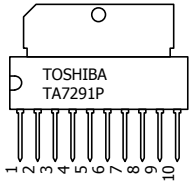
TRUTH TABLE 74541

INPUTS			OUTPUTS
$\overline{OE1}$	$\overline{OE2}$	A_n	Y_n
H	X	X	Z
X	H	X	Z
L	L	H	H
L	L	L	L

NOTES
H : High Voltage Level
L : Low Voltage Level
X : Don't Care
Z : High Impedance



BRIDGE DRIVER for DC MOTOR
TA7291P



PIN Explanation TA7291P

1	2	4	5	6	7	8	10
GND	OUT1	V_{ref}	IN1	IN2	V_{CC}	V_S	OUT2

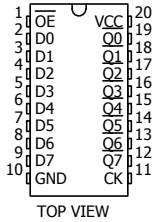
Pin (3), (9): NC

FUNCTION TA7291P

INPUTS		OUTPUTS		MODE
IN1	IN2	OUT1	OUT2	
L	L	Z	Z	STOP
H	L	H	L	CW/CCW
L	H	L	H	CCW/CW
H	H	L	L	BRAKE

NOTES
H : High Voltage Level
L : Low Voltage Level
Z : High Impedance

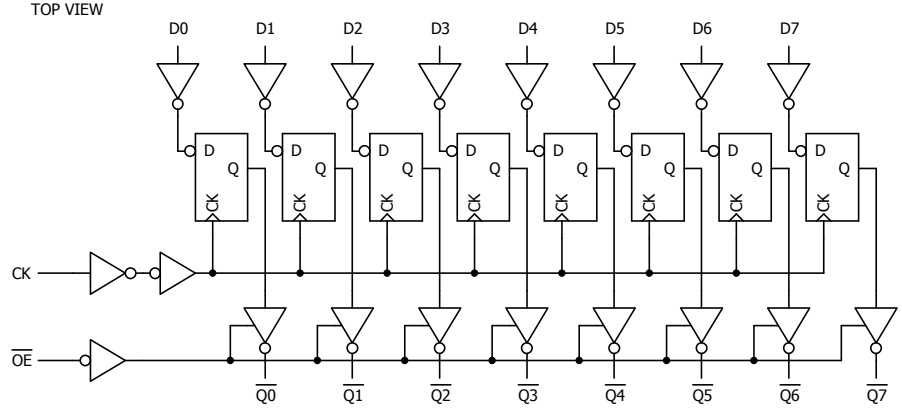
Octal D-Type Filp-Flop with 3-State Output, Inverting
74HC564



TRUTH TABLE 74564

INPUTS			OUTPUTS
$\overline{OE}$	CK	D_n	$\overline{Q_n}$
H	X	X	Z
L	↓	X	$\overline{Q_n}$
L	↑	L	H
L	↑	H	L

NOTES
H : High Voltage Level
L : Low Voltage Level
X : Don't Care
Z : High Impedance
$\overline{Q_n}$: No Change



DC MOTOR
RC-260RA-18130

DC Operating Voltage 3-6V
Speed* 13400rpm
Current* 0.13A

*6V No Load

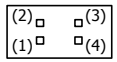
STEPPER MOTOR
42M048C1U

DC Operating Voltage 5V
Step Angle* 7.5°
Steps per Revolution* 48

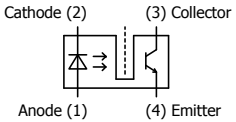
*Measured with 2 phases energized

Photointerrupter
RPI 574

TOP VIEW



BOTTOM VIEW



競技上の注意事項

1. 作業は、指示された場所で安全に行ってください。
2. 工具等の貸し借りは禁止です。
3. 練習で作成した設計製作回路やプログラムの持ち込みは禁止です。
4. 事前に作成したヘッダーファイルなどは、審査を受けてください。
5. 競技開始の合図があるまで、準備はしないでください。
   (テーブルタップの準備は競技開始前に行いますが、パソコンおよび工具等の電源プラグは抜いておいてください。)
6. 入力回路は、支給された電子部品を使って製作してください。競技中にハンダ、スズメッキ線が不足した場合には、手をあげて申し出てください。
7. プログラムは、順番どおりに作成しなくてもかまいません。
8. 予備審査の得点は、完全に動作した場合に加点されます。
9. 作成した課題プログラム(未完成も含む)のソースリストを、予備審査前に記録媒体にコピーし提出してください。また、ファイル名は表 1 を参考に付けてください。記録媒体にコピーする時間は競技時間の 2 時間 30 分以外に確保します。

表 1　制御プログラムのファイル名

課題	ファイル名の例
1	kadai-1.c　kadai-1.asm　kadai-1.txt　など
2	kadai-2.c　kadai-2.asm　kadai-2.txt　など
3	kadai-3.c　kadai-3.asm　kadai-3.txt　など
…	…

※ファイル名は課題番号が付加されていれば、p1.c などでも良い。

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
プログラム解説マニュアル

第 1.00 版

2012.02.01

ルネサスマイコンカーラリー事務局

注 意 事 項 (rev.3.1R)

著作権

・本マニュアルに関する著作権はルネサスマイコンカーラリー事務局に帰属します。
・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、文書によるルネサスマイコンカーラリー事務局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、ルネサスマイコンカーラリー事務局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、ルネサスマイコンカーラリー事務局は、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。

連絡先

(株)ルネサスソリューションズ　ルネサスマイコンカーラリー事務局
〒162-0824　東京都新宿区揚場町 2-1　軽子坂MNビル
TEL (03)-3266-8510
E-mail:official@mcr.gr.jp

目　次

1. 概要

本マニュアルは、第 11 回高校生ものづくりコンテスト全国大会の電子回路組立部門の課題について解説したマニュアルです。電子回路組立部門には、回路製作（半田付け）と課題のプログラム作成がありますが、本マニュアルでは主に、プログラムについて説明します。

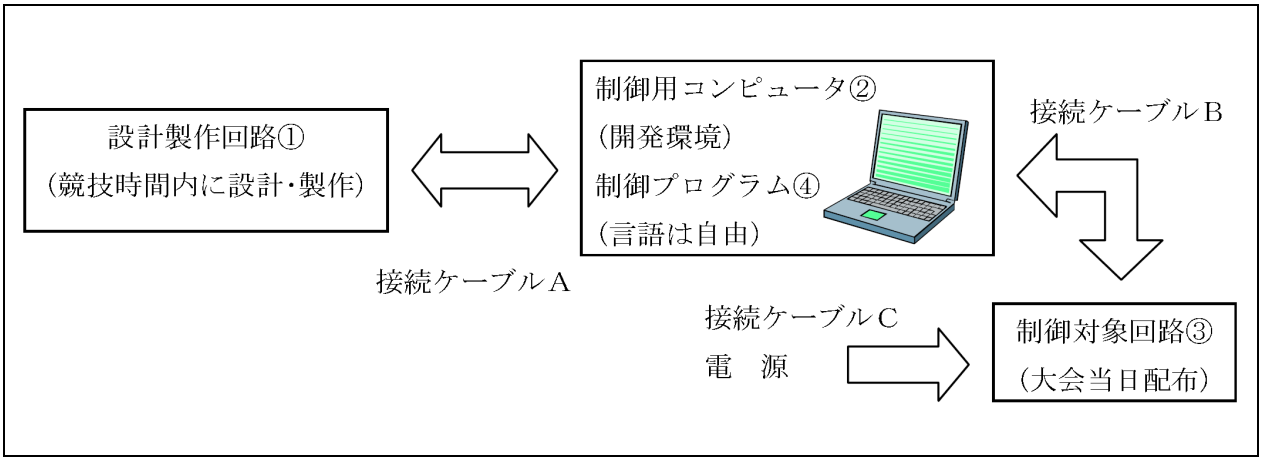
高校生ものづくりコンテストについては、全国工業高等学校長協会のホームページ（アドレス：http://www.zenkoukyo.or.jp/）を参照してください。

▲競技中の様子

2. 構成

2.1 全体構成

全体の接続構成を下図に示します。



▲事前配付資料より抜粋

設計製作回路①	支給された部品を使って、競技時間内に製作する回路です。事前配付資料には、下記のように記載されています。 **大会当日に示す設計仕様に基づく電子回路を設計し、ユニバーサル基板を用いて電子回路基板を製作する。配線はスズメッキ線を使用し、設計製作回路は以下の部品を使用する。** **①ユニバーサル基板（サンハヤト ICB293 相当品）** **②フォトインタラプタ、スイッチ、コネクタ、0.5φスズメッキ線等**

<table>
<tr><td>制御用コンピュータ②</td><td>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。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。

実際は、ルネサス エレクトロニクス製マイコン(R8C マイコンなど)、マイクロチップ・テクノロジー製マイコン(PIC マイコンなど)、ポケットコンピュータなどが使われています。ただ、ポケットコンピュータは、A/D 変換機能を使った課題が 2008 年度に出題されてからほとんど使われなくなりました。

今回の制御用コンピュータ②の構成を、下記に示します。

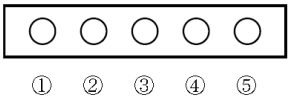
パソコン：WindowsXP、または WindowsVista、または Windows7、USB コネクタ搭載
マイコンボード：マイコンカーラリー承認ボード　RY_R8C38 ボード
開発環境：ルネサス統合開発環境
開発言語：C 言語

マイコンボードは、ジャパンマイコンカーラリーの承認ボードである「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。</td></tr>
<tr><td>制御対象回路③</td><td>大会当日に配布される基板です。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</td></tr>
<tr><td>制御プログラム④</td><td>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。</td></tr>
<tr><td>接続ケーブル A</td><td>事前配付資料で、設計製作回路①側のピン配置が決められており、下記のように記載されています。

ICピッチ１列5ピンコネクタ・メス（ストレートピンヘッダハウジング）
 
①　②　③　④　⑤
<table><tr><td>①</td><td>ＧＮＤ</td><td>②</td><td>５Ｖ</td><td>③</td><td>入力０</td></tr><tr><td>④</td><td>入力１</td><td>⑤</td><td>入力２</td><td></td><td></td></tr></table>
接続ケーブル A は、あらかじめ製作しておきます。</td></tr>
</table>

<table>
<tr><td>接続ケーブル B</td><td>事前配付資料で、制御対象回路③側のピン配置が決められており、下記のように記載されています。

ICピッチ１列１５ピンコネクタ・メス（ストレートピンヘッダ ハウジング）
○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○
① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭ ⑮

<table>
<tr><td>①</td><td>ＧＮＤ</td><td>②</td><td>出力０</td><td>③</td><td>出力１</td><td>④</td><td>出力２</td><td>⑤</td><td>出力３</td></tr>
<tr><td>⑥</td><td>出力４</td><td>⑦</td><td>出力５</td><td>⑧</td><td>出力６</td><td>⑨</td><td>出力７</td><td>⑩</td><td>出力８</td></tr>
<tr><td>⑪</td><td>出力９</td><td>⑫</td><td>ＮＣ</td><td>⑬</td><td>ＮＣ</td><td>⑭</td><td>ＮＣ</td><td>⑮</td><td>ＮＣ</td></tr>
</table>

接続ケーブル B は、あらかじめ製作しておきます。</td></tr>
<tr><td>接続ケーブル C</td><td>事前配付資料で、制御対象回路③側のピン配置が決められており、下記のように記載されています。

ICピッチ１列3ピンコネクタ・メス（ストレートピンヘッダ ハウジング）
○ ○ ○
① ② ③

<table>
<tr><td>①</td><td>ＧＮＤ</td><td>②</td><td>５Ｖ</td><td>③</td><td>ＧＮＤ</td></tr>
</table>

接続ケーブル C は、あらかじめ製作しておきます。</td></tr>
</table>

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。

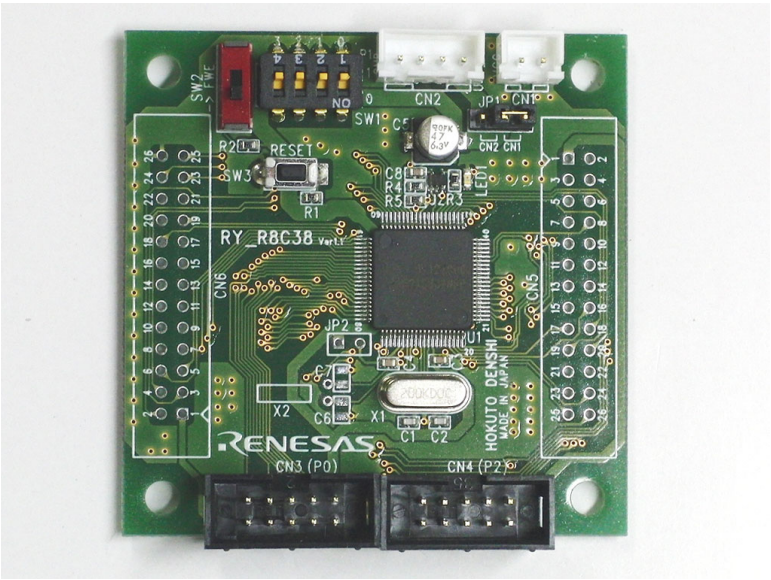
項目		時間［分］
概要の把握（ざっと読む）		5
回路図作成		5
基板製作		20
プログラム	入力回路部分のプログラム	20
	出力回路部分のプログラム	40
	課題 1～7 の main 部分のプログラム	60
合計		150

2.3　制御用コンピュータ②

本マニュアルでは、ジャパンマイコンカーラリーの承認ボードである「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サイトにある「マイコン実習マニュアル(R8C/38A 版)」を参照ください。

マイコン実習マニュアル(R8C/38A 版)は、
http://www.mcr.gr.jp/tech/download/main01.html
↓
マイコンに関する資料(R8C 編)
より、ダウンロードできます。

マイコンボードの購入方法は、マイコンカーラリー販売サイト（URL:http://www2.himdx.net/mcr/）を参照してください。



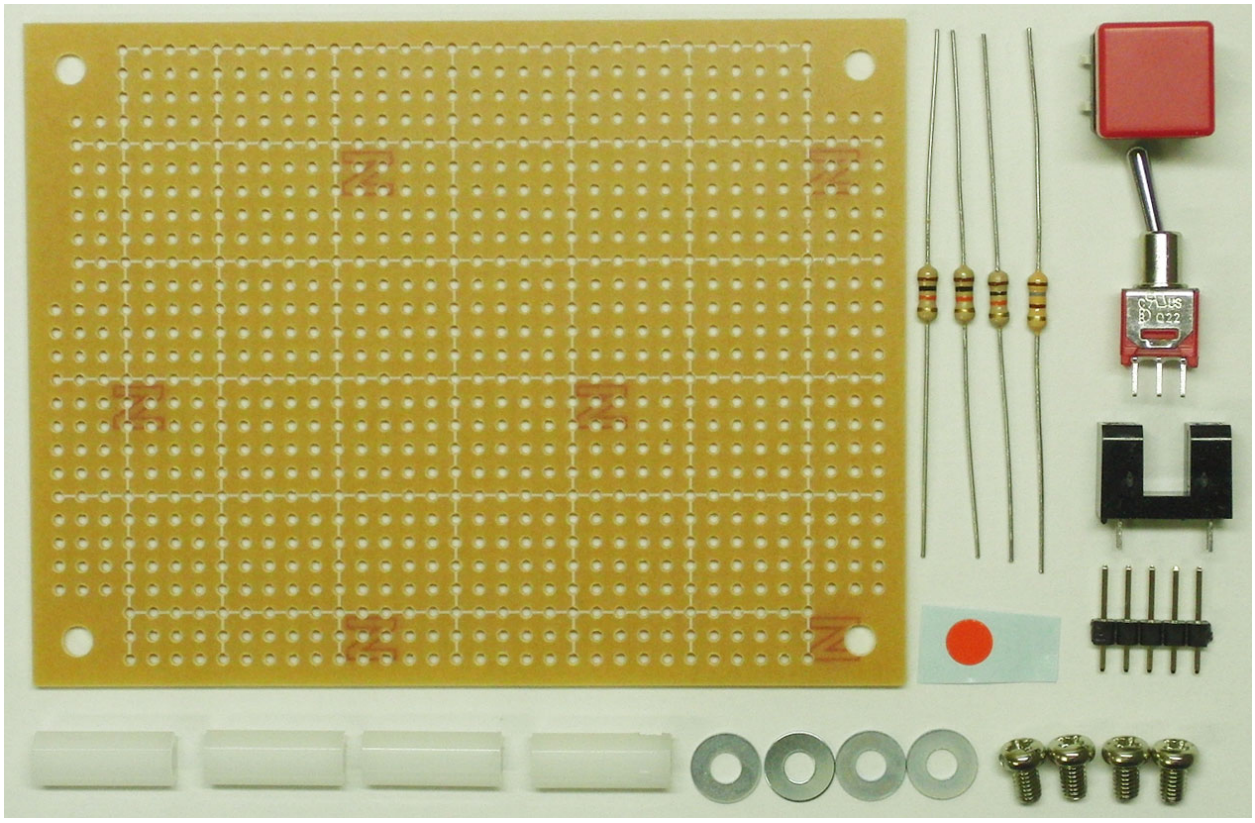
▲RY_R8C38 ボード（マイコンはルネサス エレクトロニクス製の R8C/38A）

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。支給部品を下記に示します。

番号	部品名	型番など	個数
1	ユニバーサル基板	サンハヤト ICB-293	1
2	透過型フォトインタラプタ	ROHM RPI-574	1
3	タクトスイッチ	青、黒、赤、緑、黄のいずれか	1
4	トグルスイッチ	単極双投(ON-ON)	1
5	ピンヘッダ	5 ピンストレート	1
6	抵抗	10kΩ　1/4W	3
7	抵抗	180Ω　1/4W	1
8	ネジ	M3　10mm	4
9	平ワッシャ	M3	4
10	スペーサー	15mm	4
11	赤丸シール	基板原点のマーク用	1
12	スズメッキ線	φ0.5	適量
13	鉛フリーハンダ	Sn/3Ag/0.5Cu　φ0.8　(HOZAN)	適量

▲大会当日配付資料より抜粋



▲大会当日支給された部品

5 ピンのヘッダピンの接続は指定されています。接続内容を下記に示します。

2.54 inch　1 列 5 ピンストレート

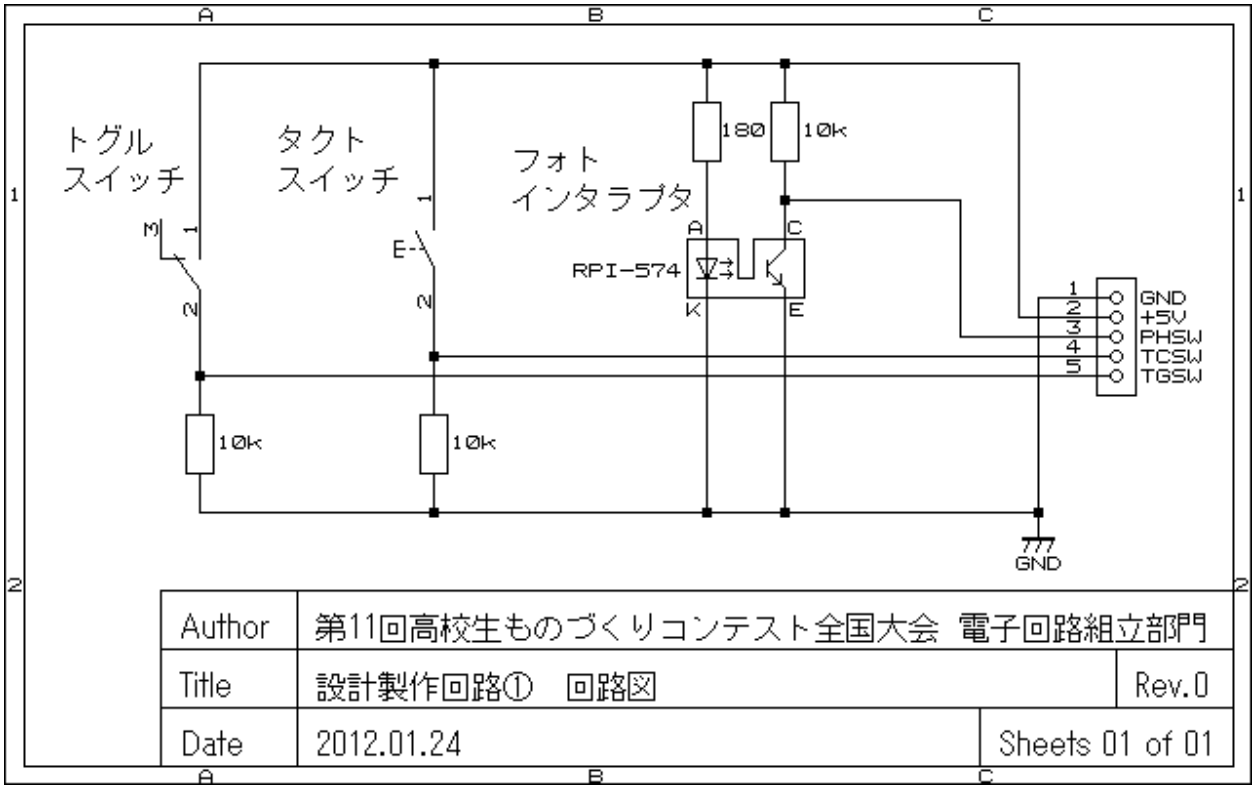
○ ○ ○ ○ ○

1　2　3　4　5

ピン番号	端子名	略称
1	GND	GND
2	+5V	5V
3	透過型フォトインタラプタ	PHSW
4	タクトスイッチ	TCSW
5	トグルスイッチ	TGSW

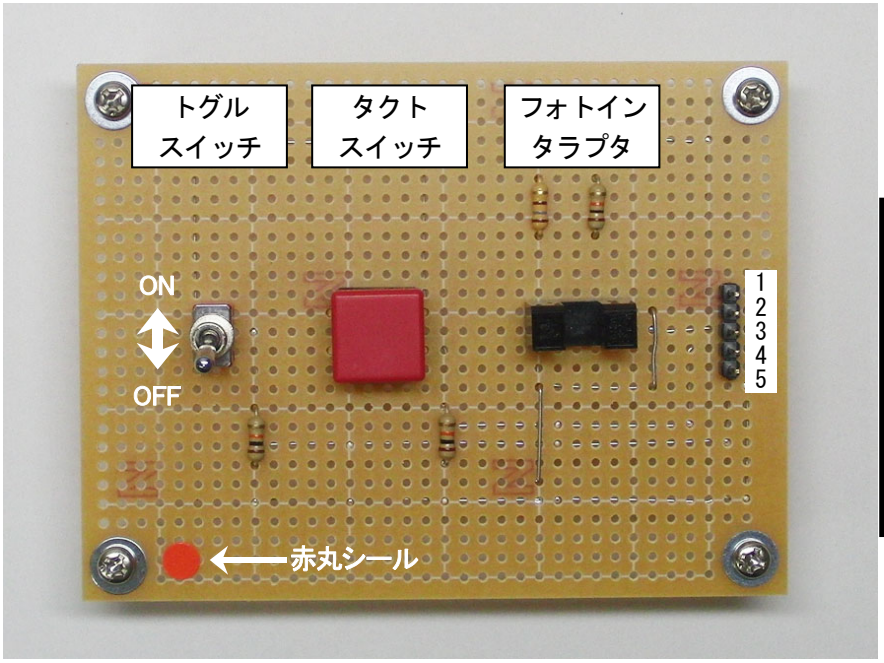
▲大会当日配付資料より抜粋

競技者は部品から回路を予想して組み立てます。
回路図の作成例を下記に示します。



▲回路図　作成例

完成した基板の製作例を下記に示します。今年の問題には、「**基板の左下に赤丸シールを張り、基板原点とする。**」という記載がありました。赤丸シールがなければ減点になります。その年限りの要求がある場合もありますので、問題をよく読んで対応してください。



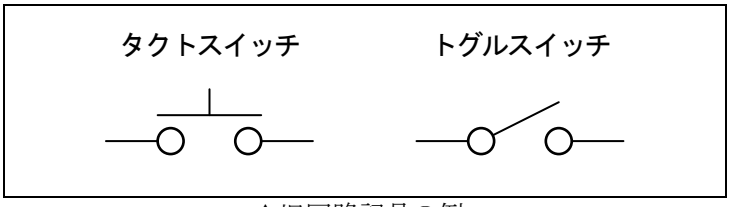
平ワッシャの位置、ナットの向きにも気をつけて組み立てましょう。細かいことですが、減点の対象となります。

▲基板製作例

■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。

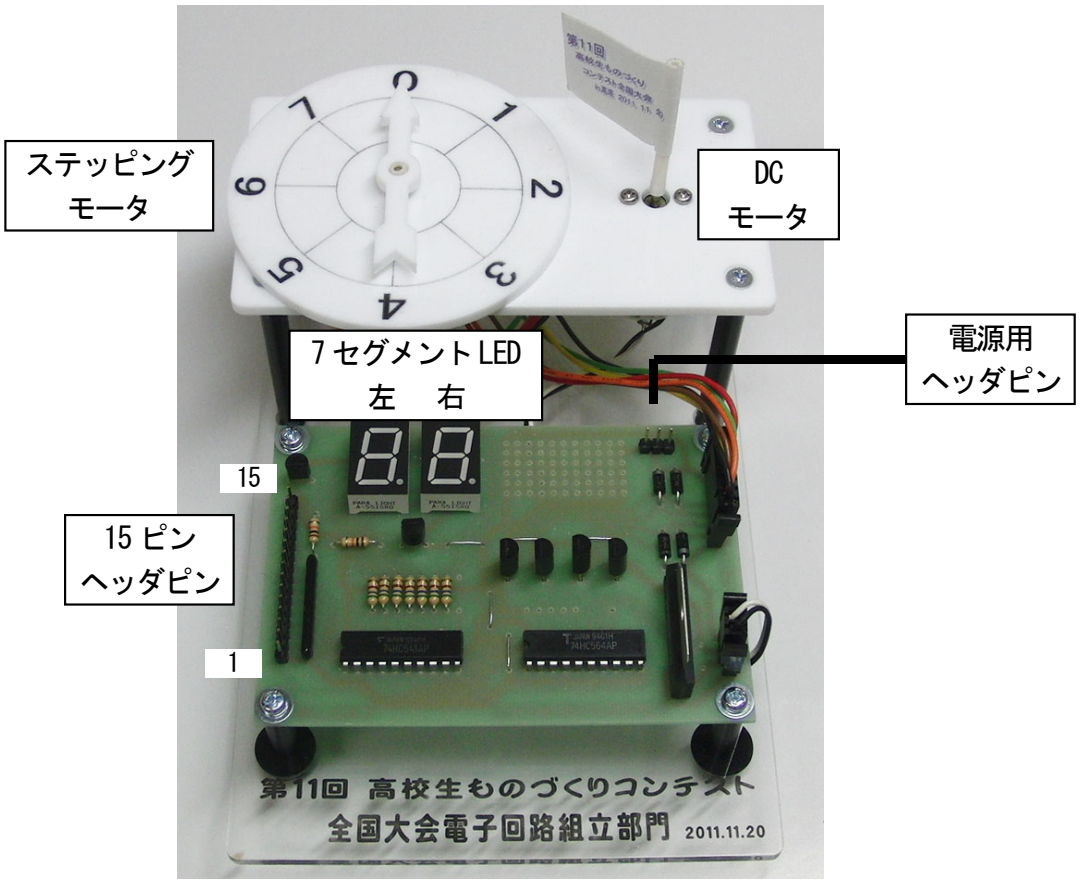
配置は、基本的に信号の流れは左から右に、電圧は高いほうが上、低いほうが下です。入出力一体回路などで必ずしも左から右に書けないこともありますが、今回は入力回路ですのでスイッチやフォトカプラは左側にあって、入力回路の出力端子が一番右側にあるのがいいと思います。回路記号は新記号（JIS C 0617、IEC 60617）に準拠しますが、まだ書籍や部品表などに旧記号（JIS C 0301）が使われている場合があるので、当面旧記号も認められています。抵抗などの数値は必須です。端子はピンの丸とそれらを囲む四角で書かれますが、ピン番号(1,2,3･･･)とピン名称（5V,GND,TCSW など）も記入しましょう。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。

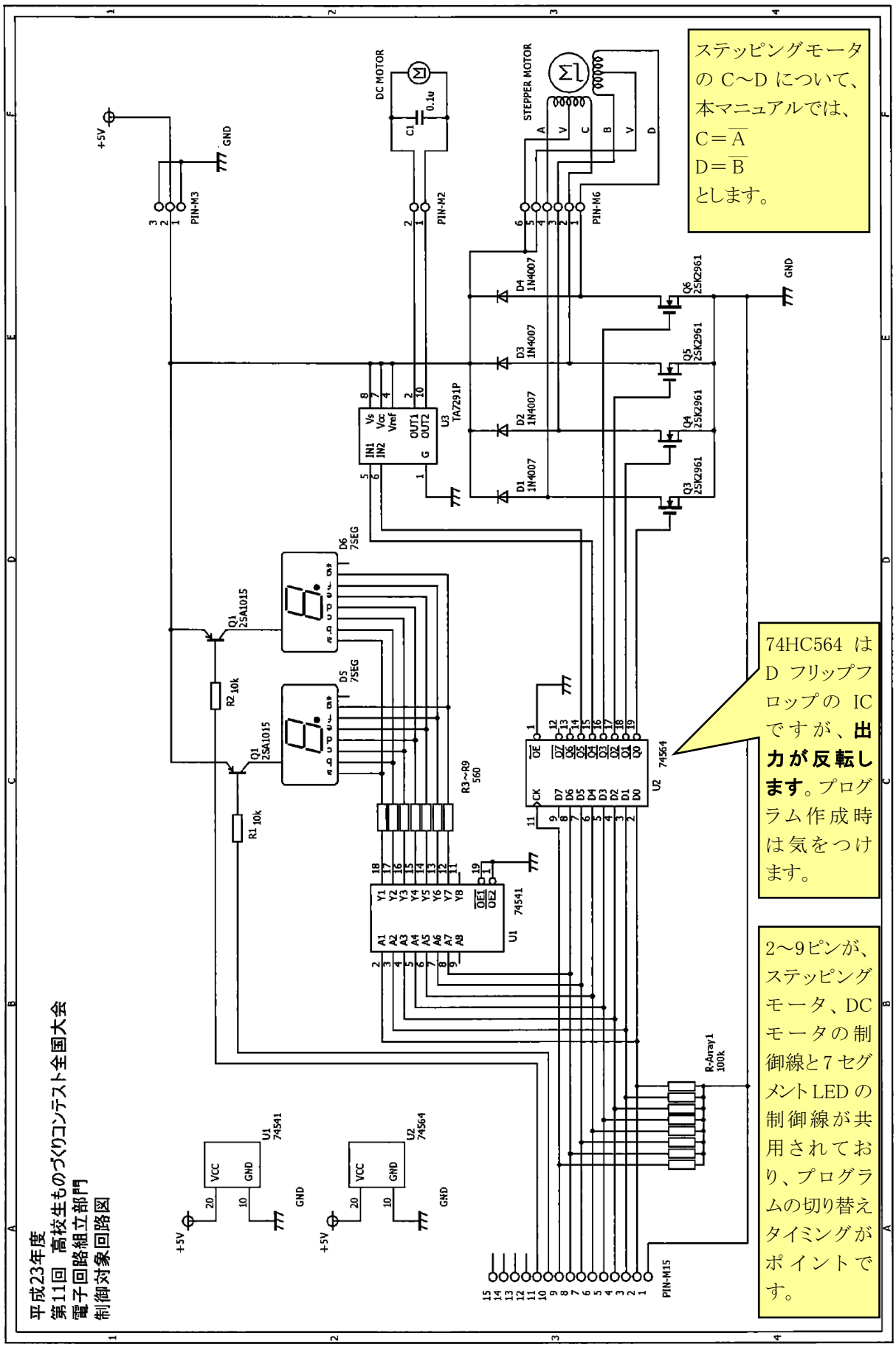


▲旧回路記号の例

2.5　制御対象回路③

制御対象回路③は、大会当日に配布される基板です。
制御対象回路③の写真を下記に、回路図を次ページに示します。



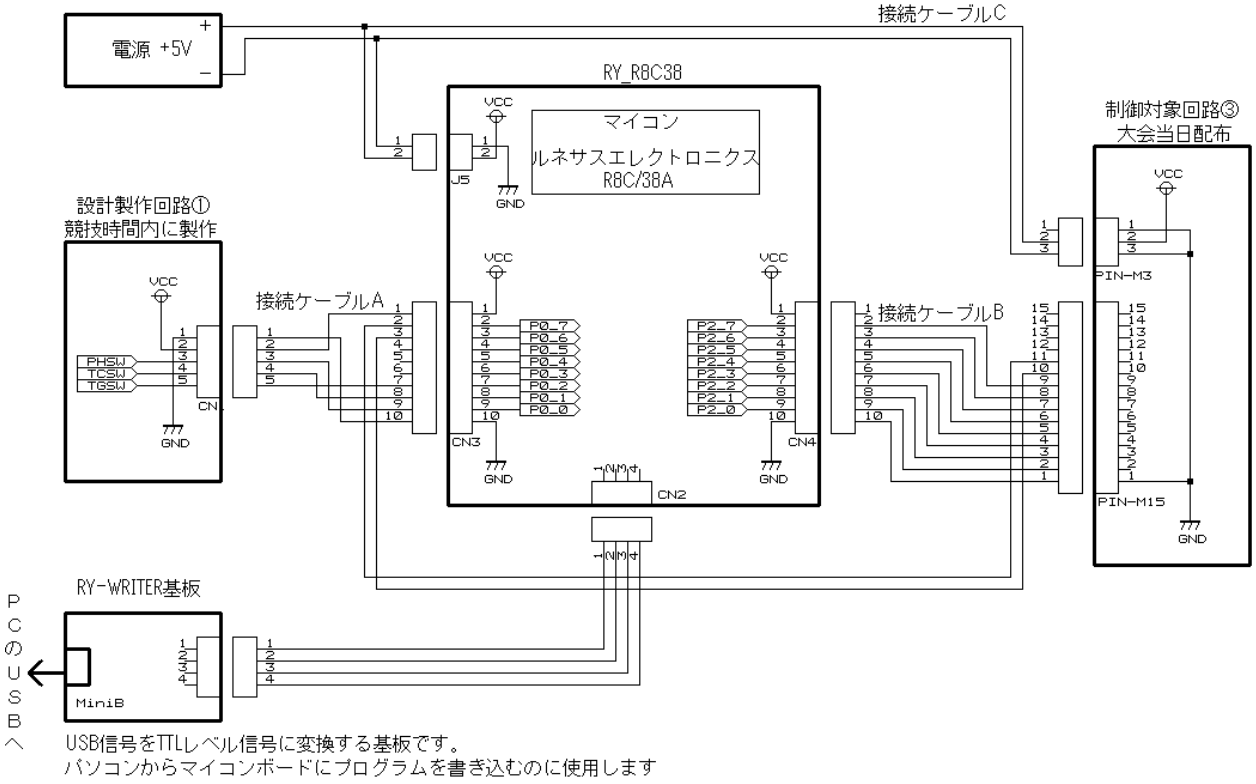

ステッピングモータの C～D について、本マニュアルでは、
C＝$\overline{A}$
D＝$\overline{B}$
とします。
74HC564 は D フリップフロップの IC ですが、**出力が反転します**。プログラム作成時は気をつけます。
2～9ピンが、ステッピングモータ、DC モータの制御線と 7 セグメント LED の制御線が共用されており、プログラムの切り替えタイミングがポイントです。
平成23年度
第11回 高校生ものづくりコンテスト全国大会
電子回路組立部門
制御対象回路図
DC MOTOR
STEPPER MOTOR
TA7291P
74541
74564
PIN-M3
PIN-M2
PIN-M6
PIN-M15
+5V
GND

▲大会当日配付資料より

2.6 接続

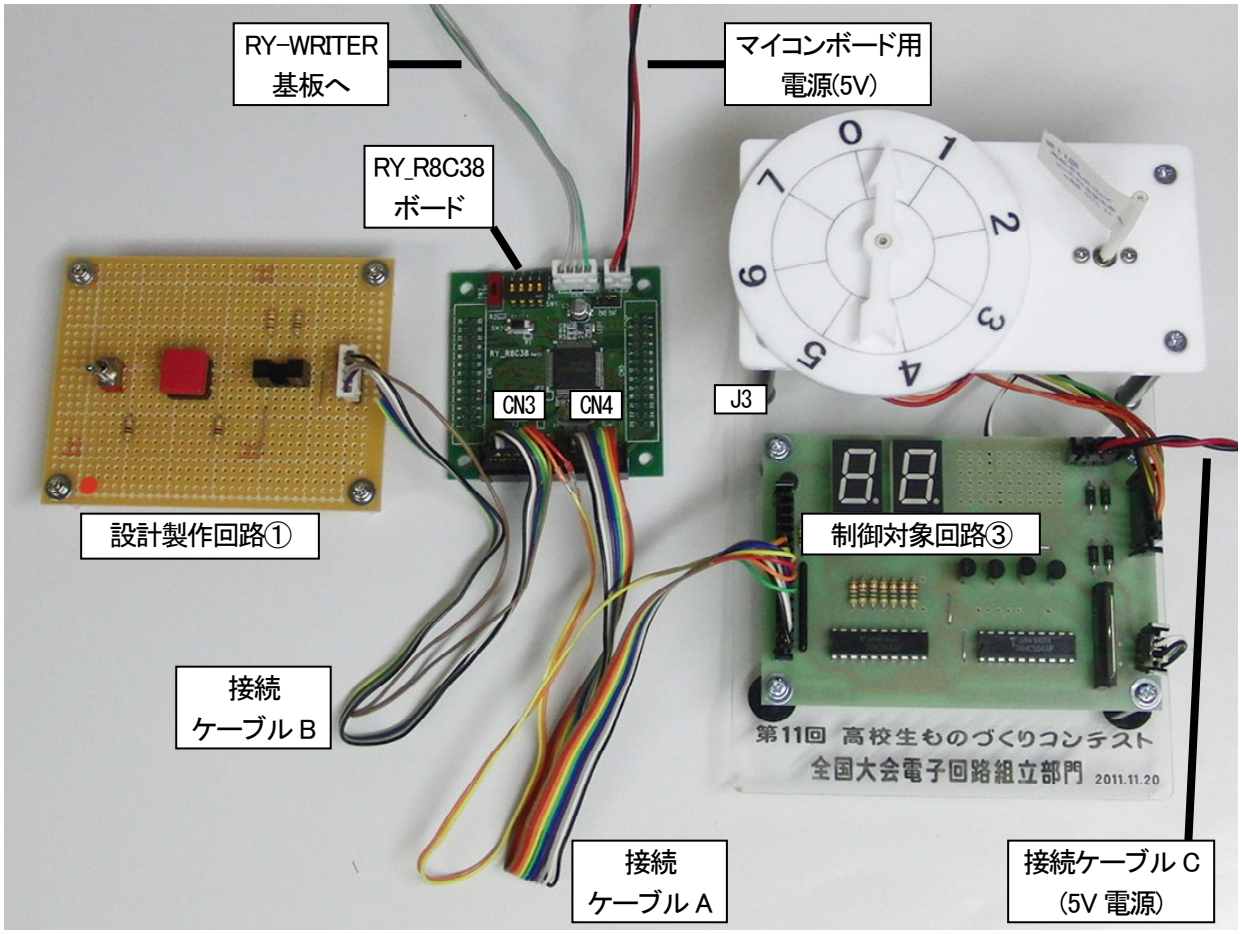
設計製作回路①（競技時間内に製作）、制御対象回路③（大会当日配布）、RY_R8C38 ボード（持ち込み）との接続を下記に示します。接続ケーブル A、接続ケーブル B、接続ケーブル C は事前に製作しておきます。

今回のプログラムは、下記のように結線されているものとします。



▲結線例

接続したときの様子（例）を下記に示します。



▲接続したときの様子（例）

3. プログラムの開き方

3.1 開発環境

本マニュアルでは、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サイトにある「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。

ルネサス統合開発環境 操作マニュアル(R8C/38A 版)は、
http://www.mcr.gr.jp/tech/download/main01.html
↓
マイコンに関する資料(R8C 編)
より、ダウンロードできます。

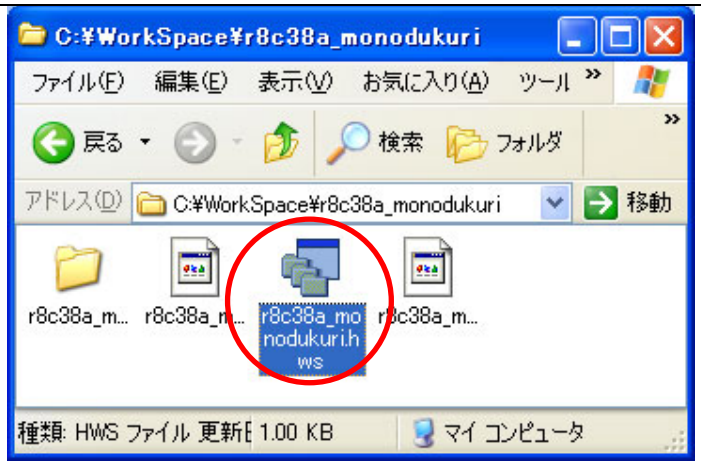
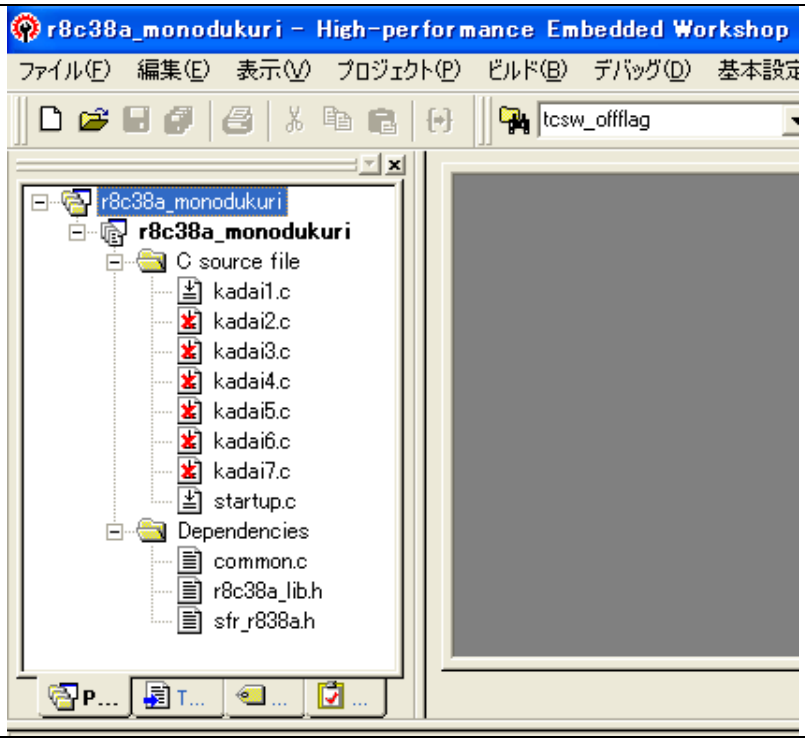
3.2 プログラムのダウンロード

「平成 23 年度　高校生ものづくりコンテスト電子回路組立部門の回答例プログラム」をダウンロードする手順を、下記に示します。

1	サンプルプログラム、書き込みソフトのダウンロード(H8編) 2010.10.07更新 サンプルプログラム、書き込みソフトのダウンロード(R8C編) 2010.10.07更新 マイコンに関する資料(H8編) 2009.05.25更新 マイコンに関する資料(R8C編) 2010.08.30更新 マイコンカー、オプションに関する資料(H8編) 2010.09.10更新 マイコンカー、オプションに関する資料(R8C編) 2010.09.01更新 実習に関する資料 2004.08.17更新 出版本に関する資料、その他資料 2010.04.01更新	http://www.mcr.gr.jp/tech/download/main01.html にアクセスします。 「出版本に関する資料、その他資料」をクリックします。
2	ダウンロード ～出版本に関する資料、その他資料～ 出版されている本に関するデータ、その他資料を掲載しています。 ●平成23年度　高校生ものづくりコンテスト電子回路組立部門の回答例プログラム 2012.03.01 全国工業高等学校長協会主催「平成23年度　高校生ものづくりコンテスト電子回路組立部門　全国大会」の回答例プログラムです。マイコンは、マイコンカーラリー承認ボードのRY_R8C38ボード(R8C/38Aマイコン)を使っています。マニュアルは、全国情報技術教育研究会のホームページに掲載されています。 →DOWNLOAD (EXE 約0.4MB)	「DOWNLOAD」をクリックして、ダウンロードします。
3		ダウンロードした「workspace_monodukuri2011.exe」を実行して、サンプルプログラムをインストールします。デフォルトでは、「c:¥workspace」フォルダにインストールされます。

3.3 プログラムを開く

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」フォルダにある、「r8c38a_monodukuri.hws」を実行します。
2		ルネサス統合開発環境が立ち上がります。 左側にあるリストが、プログラムファイルになります。 ※hws ファイルを開けないというメッセージがでた場合は、ルネサス統合開発環境の最新版をルネサスエレクトロニクスのホームページからダウンロードして、インストールしてください。

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¥r8c38a_monodukuri¥r8c38a_monodukuri¥sfr_r838a.h
r8c38a_lib.h	○	R8C/38A マイコン独自機能の関数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¥r8c38a_monodukuri¥r8c38a_monodukuri¥r8c38a_lib.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¥r8c38a_monodukuri¥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¥r8c38a_monodukuri¥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¥r8c38a_monodukuri¥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¥r8c38a_monodukuri¥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¥r8c38a_monodukuri¥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¥r8c38a_monodukuri¥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¥r8c38a_monodukuri¥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¥r8c38a_monodukuri¥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¥r8c38a_monodukuri¥kadai7.c

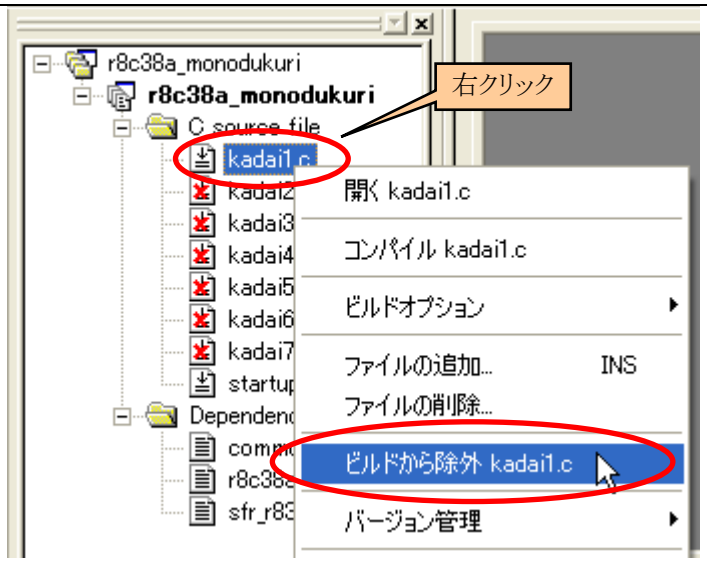
※　○は持ち込み可能　×はコンテスト時に空から作成するファイルです。

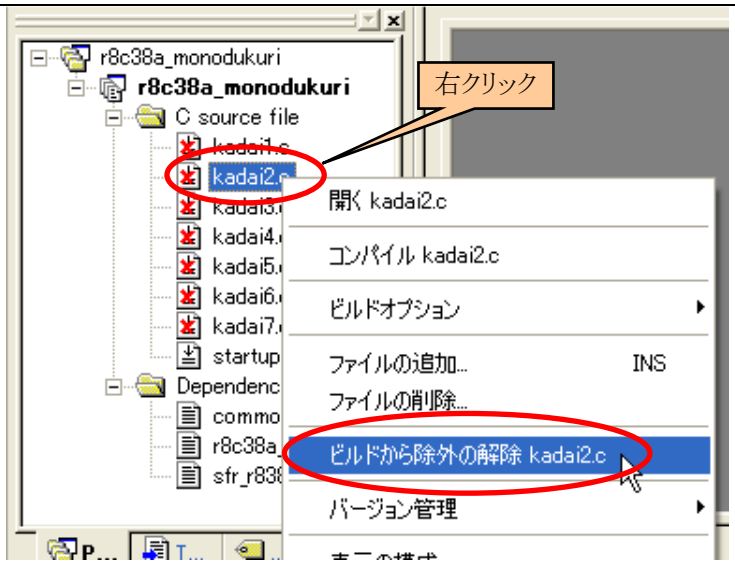
本プロジェクトには、kadai1.c～kadai7.c の課題ファイルが登録されていますが、この中で有効にできるのは 1 つだけです。例えば、課題 1 のときは、「kadai1.c」のみ有効にして、「kadai2.c～kadai7.c」はビルドから除外（ファイル左の赤い×マーク）にしておきます。

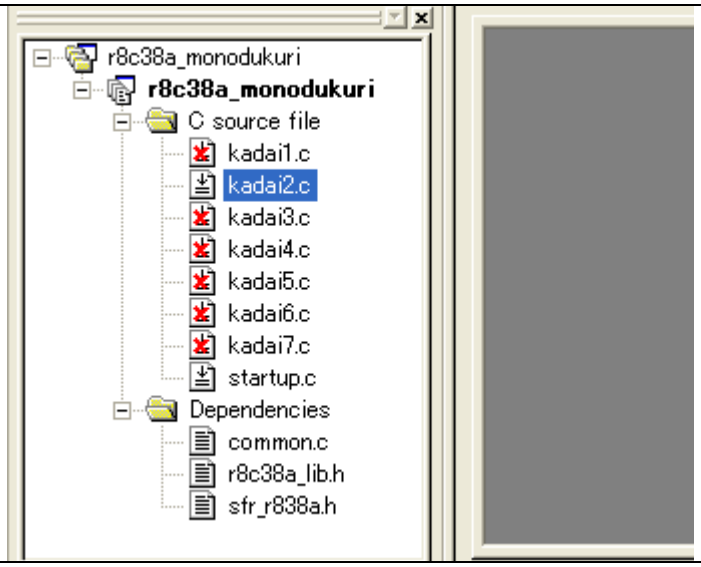
各課題のプログラムをビルドするときに、ビルドから除外するファイルを下記に示します（ファイルの詳しい構成は後述します）。

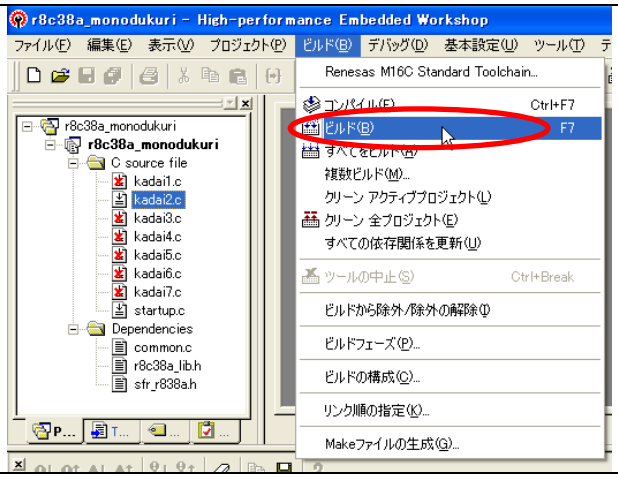
ファイル名	課題 1	課題 2	課題 3	課題 4	課題 5	課題 6	課題 7
startup.c	○	○	○	○	○	○	○
common.c	○	○	○	○	○	○	○
kadai1.c	○	×	×	×	×	×	×
kadai2.c	×	○	×	×	×	×	×
kadai3.c	×	×	○	×	×	×	×
kadai4.c	×	×	×	○	×	×	×
kadai5.c	×	×	×	×	○	×	×
kadai6.c	×	×	×	×	×	○	×
kadai7.c	×	×	×	×	×	×	○

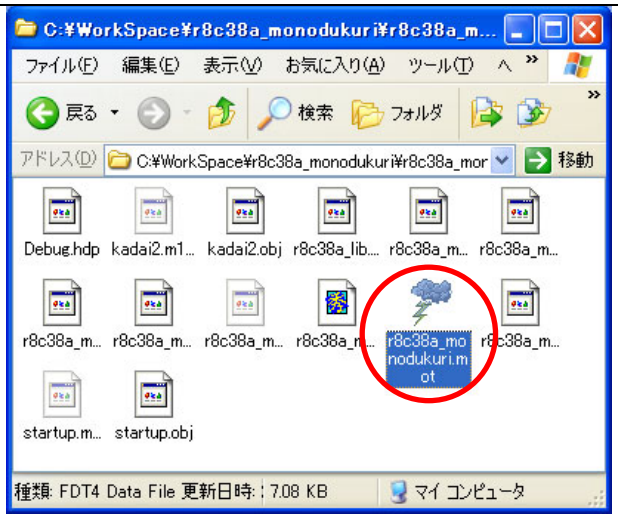
○：有効　×：ビルドから除外するファイル

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。

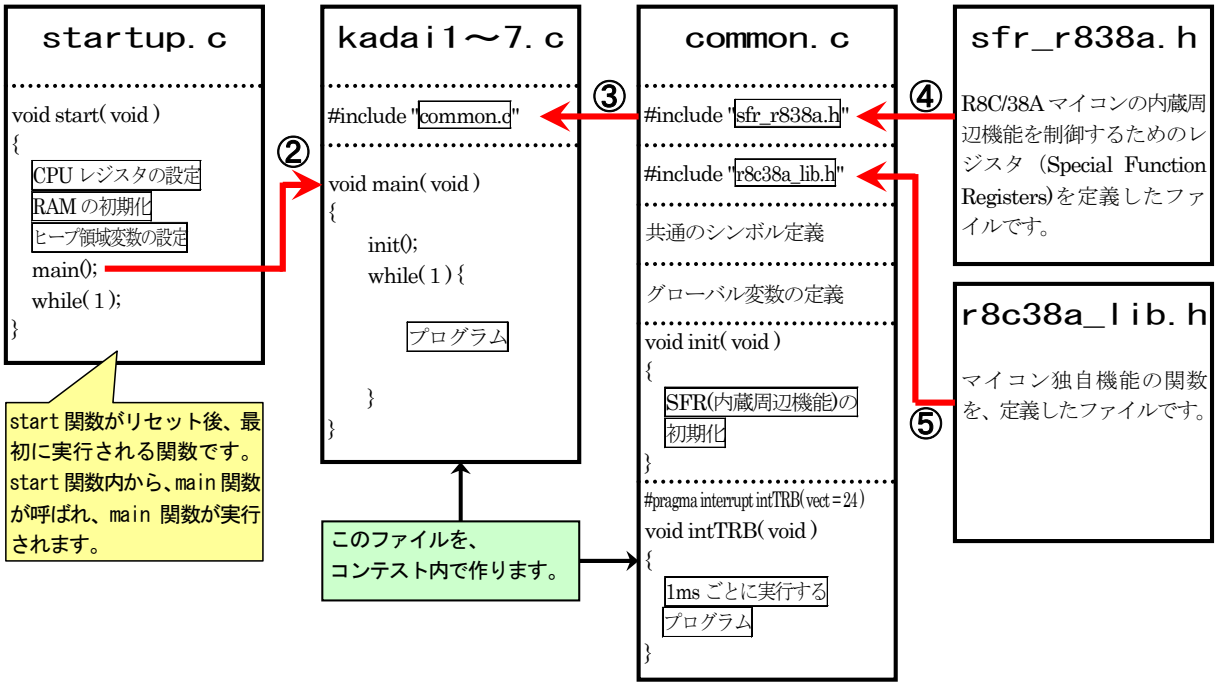
7		MOT ファイルは、 「C:¥Workspace¥r8c38a_monodukuri¥r8c_38a_monodukuri¥Debug」フォルダ内にできます。

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. プログラム

4.1 ファイル構成

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	common.c は、r8c38a_lib.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、マイコン独自機能の関数を定義したファイルです。

4.2 「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。

4.3 「common.c」ファイル－init関数

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。

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タイマ RB の設定
    ③ポートの入出力設定
    ④割り込みの許可
}
```

4.3.1 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。

```
110 :      //  CPU の動作クロックを XIN クロックにする
111 :      init_xin_clk();
```

4.3.2 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。タイマ RB の設定は、set_timer_b 関数を使います。第 1 引数はタイマ RB の動作モードを設定します。今回は定期的な割り込みとして使うので「INTERVAL_INT」を設定します。第 2 引数は、割り込み周期をμs 単位で設定します。今回は、1ms なので、1000 を設定します。

```
113 :      // タイマ RB で 1ms ごとに割り込みを発生
114 :      set_timer_b( INTERVAL_INT, 1000 );
```

※R8C/38A マイコン固有の設定をする関数を使っています。詳しくは「R8C/38A マイコン制御ライブラリ　解説マニュアル」を参照してください。

4.3.3 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 11 ピン （出力）**	**制御対象 回路③ 10 ピン （出力）**				**設計製作 回路① トグル スイッチ （入力）**	**設計製作 回路① タクト スイッチ （入力）**	**設計製作 回路① フォト インタラプタ （入力）**
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 9 ピン （出力）**	**制御対象 回路③ 8 ピン （出力）**	**制御対象 回路③ 7 ピン （出力）**	**制御対象 回路③ 6 ピン （出力）**	**制御対象 回路③ 5 ピン （出力）**	**制御対象 回路③ 4 ピン （出力）**	**制御対象 回路③ 3 ピン （出力）**	**制御対象 回路③ 2 ピン （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								

※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は、pd 関数で行います。第 1 引数はポート番号、第 2 引数は入出力方向を決める値です。入出力方向を決める値は、「入力にしたいビットは"0"、出力にしたいビットは"1"」を設定します。未接続のビットの端子は出力にしておきます。今回の入出力設定プログラムを、下記に示します。

```
116 :      // ポートの入出力設定
117 :      pd( 0 ,0xf8 );                    // 7-6:回路③ 2-0:回路①
118 :      pd( 1, 0xf0 );                    // 5:RXD0 4:TXD0 3-0:DIP SW
119 :      pd( 2 ,0xff );                    // 7-0:回路③
120 :      pd( 3 ,0xff );
121 :      pd( 4 ,0xb8 );                    // 7:XOUT 6:XIN 5:LED 2:VREF
122 :      pd( 5 ,0xff );
123 :      pd( 6 ,0xff );
124 :      pd( 7 ,0xff );
125 :      pd( 8 ,0xff );
126 :      pd( 9 ,0xff );
```

4.3.4 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。

```
128 :      // 割り込み許可
129 :      ei();
```

4.4 「common.c」ファイルの割り込みプログラム－概要

init 関数で設定したタイマ RB 割り込みの割り込みプログラムについて説明します。プログラムを下記に示します。

```
#pragma interrupt intTRB( vect = 24 )
void intTRB( void )
{
    ①時間を測定するプログラム
    ②入力信号を取り込むプログラム
    ③出力回路へ信号を出力するプログラム
}
```

「#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。

①時間を測定するプログラム

1 秒待つ、などの時間を計るために変数の値をインクリメント（＋1）するプログラムを記述します。

②入力信号を取り込むプログラム

フォトインタラプタ、タクトスイッチ、トグルスイッチの情報を読み込むプログラムを記述します。

③出力回路へ信号を出力するプログラム

7 セグメント LED の左桁、右桁、ステッピングモータ、DC モータを制御するためのプログラムを記述します。

①～③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。

4.5 「common.c」ファイルの割り込みプログラム－時間の測定

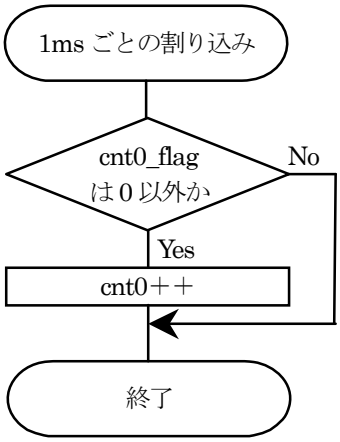
課題によっては、タクトスイッチを押した時間を計る問題などがあります。そのため、専用の変数を作り、1ms ごとに＋1するようにして、この変数を確認することで時間を計るようにします。

(1) 使用する変数

時間の測定で使用する変数を、下記に示します。

変数	内容
cnt0_flag	cnt0 変数を 1ms ごとに＋1するかしないかを設定します。 0:cnt0 の値を変化させない 1:cnt0 を 1ms ごとに＋1する
cnt0	cnt0_flag が 1 なら、1ms ごとに＋1します。cnt0_flag が 0 なら値は変化しません。

(2) フローチャート



(3) プログラム

```
// グローバル変数の宣言

  33 :  ////////////////////////////////////////////////////////////
  34 :  // カウンタ
  35 :  long    cnt0;                          // 1ms ごとに+1
  36 :  int     cnt0_flag;                     // 0:cnt0 のカウントを停止

// プログラム

 135 :  #pragma interrupt intTRB( vect = 24 )
 136 :  void intTRB( void )
 137 :  {
 138 :      // カウンタカウントアップ
 139 :      if( cnt0_flag != 0 ) {
 140 :          cnt0++;
 141 :      }
中略
 252 :  }
```

4.6 「common.c」ファイルの割り込みプログラム－入力信号の取り込み

入力信号は、設計製作回路①基板からの信号で、フォトインタラプタ、タクトスイッチ、トグルスイッチの 3 種類です。

4.6.1 入力信号のポート

各入力回路と R8C/38A マイコンのポートとの接続を、下記に示します。

入力回路	マイコンのポート	″1″が入力される条件	″0″が入力される条件	プログラムでの論理
フォトインタラプタ（PHSW）	p0_0	光りが遮断している状態（フォトインタラプタの間に障害物がある状態）	光りが透過している状態（フォトインタラプタの間に障害物が無い状態）	論理はそのままで使います。 ″1″…遮断 ″0″…透過
タクトスイッチ（TCSW）	p0_1	押している状態（ON の状態）	離している状態（OFF の状態）	論理はそのままで使います。 ″1″…ON ″0″…OFF
トグルスイッチ（TGSW）	p0_2	上側の状態（ON の状態）	下側の状態（OFF の状態）	論理はそのまま使います。 ″1″…ON ″0″…OFF

4.6.2 課題での使われ方

入力信号を取り込むためのプログラムを作り始める前に、課題 1～7 が何を要求しているのか把握します。限られた時間しかありませんので早く main 関数内のプログラムを作りたいとは思いますが、入力信号を取り込むプログラム部分を作り誤ると main 関数のプログラムが大変になりますので、この部分はある程度時間をかけましょう。

課題 1～7 で操作する内容を、下表に示します。

	フォトインタラプタ(PHSW)	タクトスイッチ(TCSW)	トグルスイッチ(TGSW)
課題 1	遮断、透過	ON、OFF	
課題 2		ON、OFF	ON、OFF
課題 3	遮断、透過	ON、OFF	
課題 4			ON、OFF
課題 5		ON→OFF で (OFF の瞬間に)	
課題 6		ON、OFF (ON にしてから OFF にする までの時間を検出)	
課題 7	・遮断された瞬間を検出する	ON、OFF	
課題 1～7 の状態をまとめると	・遮断を検出する ・透過を検出する ・遮断された瞬間を検出する	・ON を検出する ・OFF を検出する ・OFF(離したとき)の瞬間を検出 ・ON を押し続けた時間を検出	・ON を検出する ・OFF を検出する
プログラムの処理	・遮断を検出する ・透過を検出する ・遮断された瞬間を検出する	・ON を検出する ・OFF を検出する ・OFF の瞬間を検出 ※課題 6 の ON を押し続けた時間は、main 関数内で処理します。	・ON を検出する ・OFF を検出する

フォトインタラプタ、タクトスイッチ、トグルスイッチの状態を検出するプログラムは、1ms ごとに実行される割り込み処理に入れます。ただし、検出は 10ms ごとに行います。

フォトインタラプタやタクトスイッチの状態が変化した瞬間の検出は、瞬間を検出したときに 1 になる専用の変数を用意して、割り込みプログラム内で検出します。筆者としては、割り込みプログラムが少し複雑になりますが、それ以上に main 関数が簡単になるので、プログラム作成時間は短くなると考えています。

4.6.3　10msごとの処理

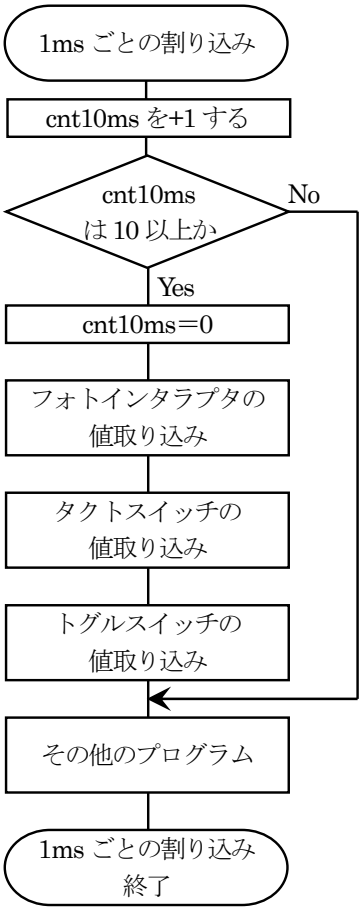
フォトインタラプタ、タクトスイッチ、トグルスイッチの状態は、10ms ごとにプログラムで読み込んでいます。
割り込みは 1ms ごとに発生するので、10 回に 1 回処理すれば、10ms ごとに処理することになります。

(1) 変数

10ms ごとの処理で使用する変数を、下記に示します。

変数	内容
cnt10ms	1ms ごとに＋1して、10 になったら 10ms たったと判断します。

(2) フローチャート



(3) プログラム

```
// グローバル変数の宣言

  38 :  int     cnt10ms;                        // 10ms ごとの処理用

// プログラム

 132 :  ////////////////////////////////////////////////////////////////////////
 133 :  // タイマ RB 1ms ごとの割り込み処理
 134 :  ////////////////////////////////////////////////////////////////////////
 135 :  #pragma interrupt intTRB( vect = 24 )
 136 :  void intTRB( void )
 137 :  {
中略
 143 :      // フォトインタラプタ、スイッチは 10ms ごとにチェックする
 144 :      cnt10ms++;
 145 :      if( cnt10ms >= 10 ) {
 146 :          cnt10ms = 0;

                フォトインタラプタの値取り込み

                タクトスイッチの値取り込み処理

                トグルスイッチの値取り込み
            }

            その他のプログラム

 252 :  }
```

4.6.4 フォトインタラプタの値取り込み処理

(1) 変数

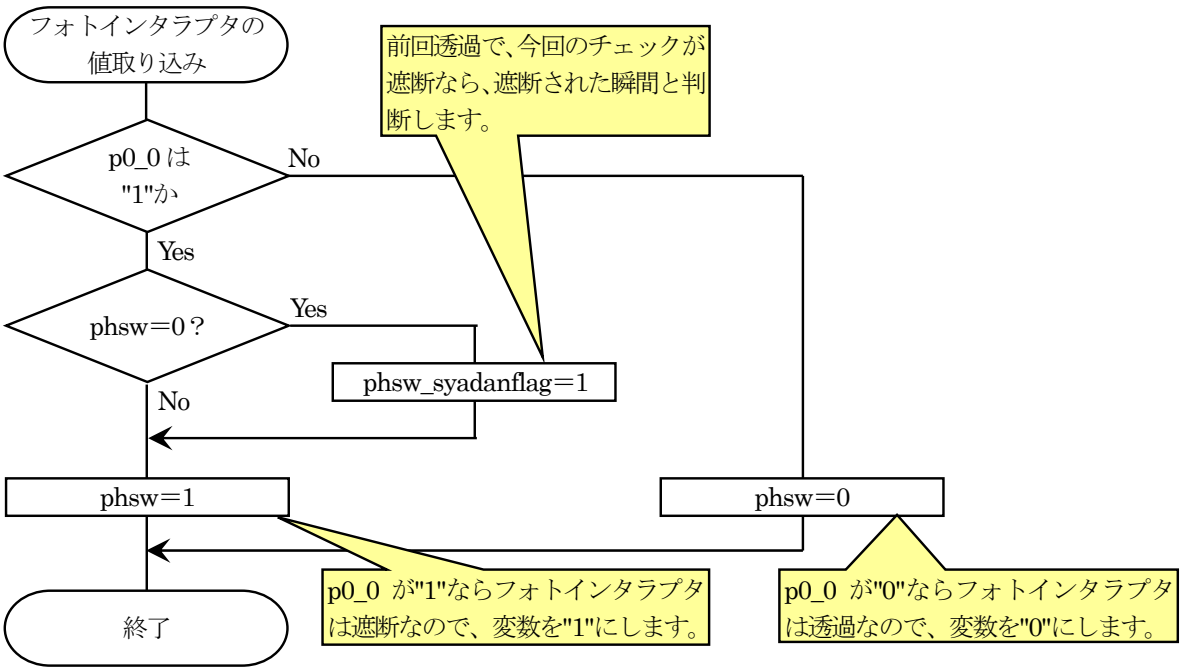
フォトインタラプタ関連で使用する変数を、下記に示します。

変数	内容

phsw	フォトインタラプタの状態が入っている変数です。
	0:フォトインタラプタ透過
	1:フォトインタラプタ遮断
	
	例)
	`if( phsw == 1 ) {  // フォトインタラプタが遮断なら`
	`  プログラム`
	`}`

phsw_syadanflag	フォトインタラプタが透過から遮断された瞬間に 1 になる変数です。この変数が 1 になったら、フォトインタラプタが遮断されたと判断します。
	if 文などで phsw_syadanflag 変数が 1 になったことを検出したら、プログラムで phsw_syadanflag 変数を 0 にしておきます。0 にしないと 1 のままなので、ずっと「遮断された瞬間」と判断してしまいます。
	
	例)
	`if( ps_syadanflag == 1 ) {`
	`  ps_syadanflag = 0;  // 1 を検出したので次のチェックに備え 0 にしておく`
	`  プログラム`
	`}`

(2) フローチャート



(3) プログラム

```
// グローバル変数の宣言

  40 :  ////////////////////////////////////////////////////////////////
  41 :  // フォトインタラプタの状態
  42 :  int     phsw;                          // 0:透過 1:遮断
  43 :  int     phsw_syadanflag;               // 透過→遮断された瞬間に 1

// プログラム

 148 :          // フォトインタラプタの値取り込み
 149 :          if( p0_0 == 1 ) {              // 遮断なら
 150 :              if( phsw == 0 ) {
 151 :                  phsw_syadanflag = 1;   // 遮断した瞬間なら 1
 152 :              }
 153 :              phsw = 1;
 154 :          } else {                       // 透過なら
 155 :              phsw = 0;
 156 :          }
```

4.6.5 タクトスイッチの値取り込み処理

(1) 変数

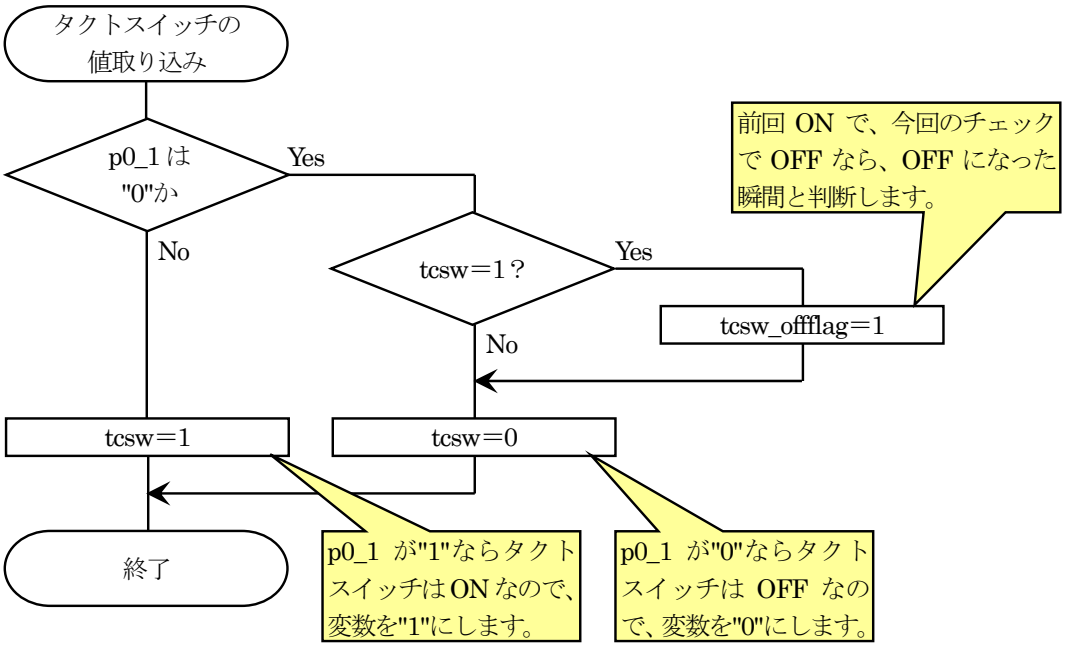
タクトスイッチ関連で使用する変数を、下記に示します。

変数	内容

tcsw	タクトスイッチの状態が入っている変数です。
	0:タクトスイッチ OFF
	1:タクトスイッチ ON
	
	例)
	`if( tcsw == 0 ) {  // タクトスイッチが OFF なら`
	`  プログラム`
	`}`

tcsw_offflag	タクトスイッチが ON から OFF になった瞬間に 1 になる変数です。この変数が 1 になったら、タクトスイッチが離されたと判断します。
	if 文などで tcsw_offflag 変数が 1 になったことを検出したら、プログラムで tcsw_offflag 変数を 0 にしておきます。0 にしないと 1 のままなので、ずっと「OFF になった瞬間」と判断してしまいます。
	
	例)
	`if( tcsw_offflag == 1 ) {`
	`  tcsw_offflag = 0;  // 1 を検出したので次のチェックに備え 0 にしておく`
	`  プログラム`
	`}`

(2) フローチャート



(3) プログラム例

プログラムを、下記に示します。

```
// グローバル変数の宣言

 45 :  ////////////////////////////////////////////////////////////////
 46 :  // タクトスイッチの状態
 47 :  int     tcsw;                          // 0:OFF 1:ON
 48 :  int     tcsw_offflag;                  // ON→OFF になった瞬間に 1

// プログラム

158 :          // タクトスイッチの値取り込み
159 :          if( p0_1 == 0 ) {              // OFF なら
160 :              if( tcsw == 1 ) {
161 :                  tcsw_offflag = 1;      // OFF になった瞬間なら 1
162 :              }
163 :              tcsw = 0;
164 :          } else {                       // ON なら
165 :              tcsw = 1;
166 :          }
```

4.6.6 トグルスイッチの値取り込み処理

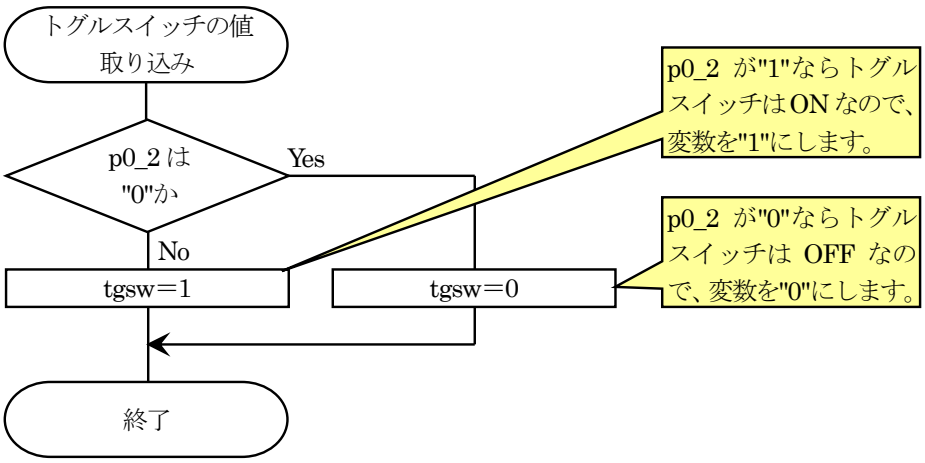
(1) 変数

トグルスイッチ関連で使用する変数を、下記に示します。

変数	内容

tgsw	トグルスイッチの状態が入っている変数です。
	0:トグルスイッチ OFF
	1:トグルスイッチ ON
	
	例)
	`if( tgsw == 0 ) {  // トグルスイッチが OFF なら`
	`  プログラム`
	`}`

(2) フローチャート



(3) プログラム

```
// グローバル変数の宣言
  50 :  ////////////////////////////////////////////////////////////
  51 :  // トグルスイッチの状態
  52 :  int     tgsw;                           // 0:OFF 1:ON

// プログラム
 168 :          // トグルスイッチの値取り込み
 169 :          if( p0_2 == 0 ) {               // OFF なら
 170 :              tgsw = 0;
 171 :          } else {                        // ON なら
 172 :              tgsw = 1;
 173 :          }
```

4.7 「common.c」ファイルの割り込みプログラム－出力回路へ信号を出力

出力信号は、制御対象回路③基板へ出力する信号で、制御する対象は左側 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 種類です。

4.7.1 出力信号のポート

各出力回路と 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
左側7セグメントLED	アノードコモン端子の選択	p0_6	全消灯	p2_6～P2_0 の値に従って点灯	7 セグメント LED はアノードコモンで、このコモン端子を ON/OFF するポートです。p0_6=″1″なら各セグメント LED は p2_6～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_6(g セグメント)	消灯	点灯	p0_6=″0″（点灯）のとき、p2_6～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_6=″0″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_5(f セグメント)	消灯	点灯	
		p2_4(e セグメント)	消灯	点灯	
		p2_3(d セグメント)	消灯	点灯	
		p2_2(c セグメント)	消灯	点灯	
		p2_1(b セグメント)	消灯	点灯	
		p2_0(a セグメント)	消灯	点灯	
右側7セグメントLED	アノードコモン端子の選択	p0_7	全消灯	p2_6～P2_0 の値に従って点灯	7 セグメント LED はアノードコモンで、このコモン端子を ON/OFF するポートです。p0_7=″1″なら各セグメント LED は p2_6～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_6(g セグメント)	消灯	点灯	p0_7=″0″（点灯）のとき、p2_6～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_7=″0″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_5(f セグメント)	消灯	点灯	
		p2_4(e セグメント)	消灯	点灯	
		p2_3(d セグメント)	消灯	点灯	
		p2_2(c セグメント)	消灯	点灯	
		p2_1(b セグメント)	消灯	点灯	
		p2_0(a セグメント)	消灯	点灯	

（次のページへ続く）

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
ステッピングモータ	74HC564 出力端子	p2_7	″0″→″1″になった瞬間に p2_6～p2_0 の信号が、ステッピングモータ、DC モータに出力される	変化なし	″1″にした瞬間に 74HC564 の出力端子から p2_5～p2_0 の値が出力され、ステッピングモータ、DC モータが動作します。″1″にした後は、74HC564 によって出力値が保持されるので p2_5～p2_0 の値を変化させてもモータの動作は変わりません。
	励磁コイルの選択	p2_3($\overline{B}$)	$\overline{B}$コイルを励磁する	$\overline{B}$コイルを励磁しない	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 相励磁、2 相励磁だと 1 回の励磁で 7.5 度動きます。今回は、課題 5 で 1-2 相励磁にする必要があるので、全課題 1-2 相励磁で制御します。1-2 相励磁だと 1 ステップは、3.75 度（7.5 度の半分）進むので、 1 周は 360÷3.75＝96 パルスです。 ※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。
		p2_2($\overline{A}$)	$\overline{A}$コイルを励磁する	$\overline{A}$コイルを励磁しない	
		p2_1(B)	B コイルを励磁する	B コイルを励磁しない	
		p2_0(A)	A コイルを励磁する	A コイルを励磁しない	
DCモータ	74HC564 出力端子	p2_7	″0″→″1″になった瞬間に p2_6～p2_0 の信号が、ステッピングモータ、DC モータに出力される	変化なし	ステッピングモータ部分を参照してください。
	DC モータの動作選択	p2_5、p2_4	″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		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。

74HC564 は、入力信号を反転して出力します。今回のプログラムでは反転しないものと考え、ポート 2(p2)に値を出力するときに、「~(チルダ)」で反転出力させるようにします。ただし、クロック線(p2_7)は変化させない(″0″のまま)ように気をつけます。

4.7.2 課題での使われ方

制御対象へ信号を出力するプログラムを作り始める前に、課題は何を要求しているのか把握します。限られた時間しかありませんので早く main 関数内のプログラムを作りたいとは思いますが、制御対象へ信号を出力するプログラム部分を作り誤ると main 関数のプログラムが大変になりますので、この部分はある程度時間をかけましょう。

課題 1～7 で制御する内容を、下表に示します。

※7 セグメント LED の消灯、ステッピングモータの停止、DC モータの停止は省略

	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
課題 1	・ON 表示 ・OFF 表示	・ON 表示 ・OFF 表示		
課題 2			・時計回りに回転 ・反時計回りに回転	
課題 3				・**低速回転で**反時計回りに回転 ・**高速回転で**反時計回りに回転
課題 4	・－表示 ・0、1 表示	・－表示 ・0～9表示		
課題 5		・0～C 表示	・時計回りに回転 (回転は、3.75 度ごと) **※7 セグメント LED の表示とモータのステップ数を合わせる制御が必要**	
課題 6			・反時計回りに回転	・反時計回りに回転
課題 7		・0～5 表示	・時計回りに回転 ・反時計回りに回転 **※7 セグメント LED の表示とモータのステップ数を合わせる制御が必要**	
課題 1～7 の状態をまとめると	・－表示 ・ON、OFF 表示 ・0、1 表示	・－表示 ・ON、OFF 表示 ・0～C 表示	・反時計回りに回転 ・時計回りに回転	・**低速回転で**反時計回りに回転 ・**高速回転で**反時計回りに回転 ・反時計回りに回転
プログラムの処理	・－表示 ・ON、OFF 表示 ・0～F 表示	・－表示 ・ON、OFF 表示 ・0～F 表示	・時計回りに回転 ・反時計回りに回転 ・ステップ数をカウントする変数を用意する（正転時はプラス、逆転時はマイナスするようにする）	・時計回りに回転 ・反時計回りに回転 **※低速、高速は main 関数内で処理することにします**

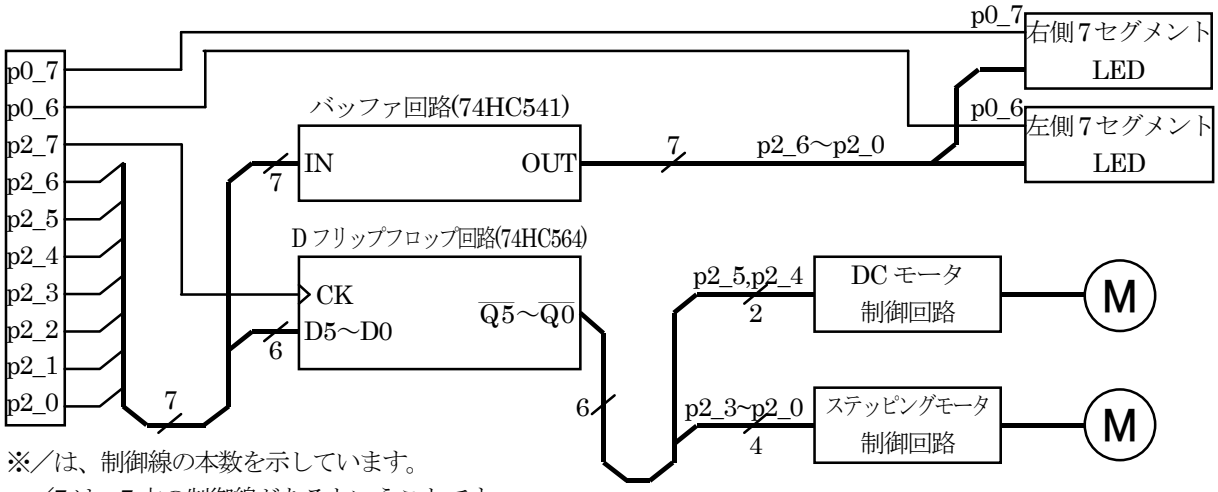
4.7.3 制御手順

制御対象回路③は特に難しい回路ではありませんが、p2_6～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、DC 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。

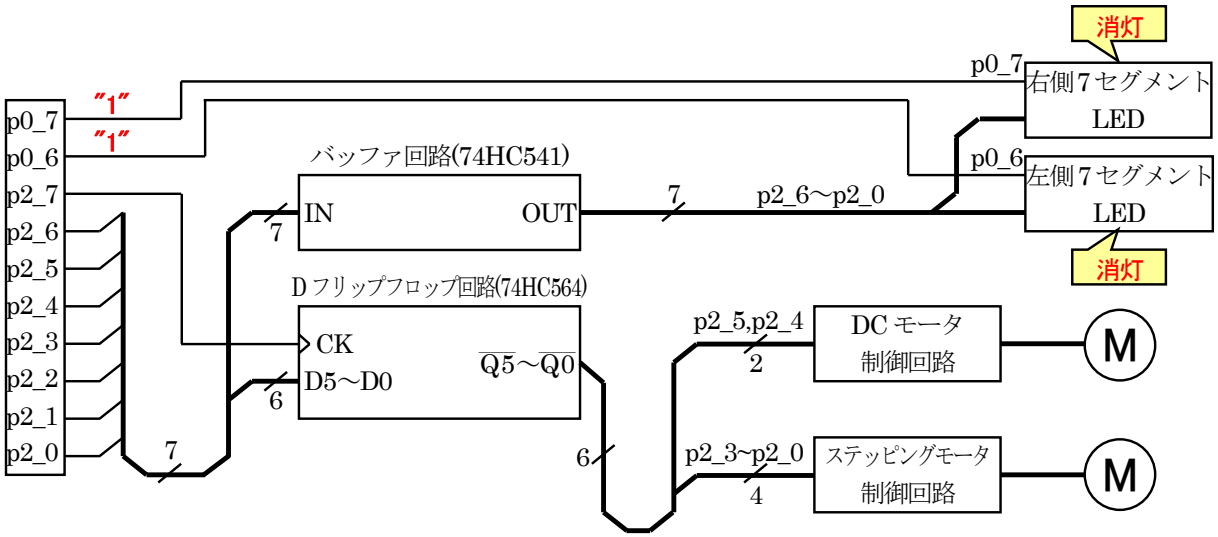
回路の簡略図、動作を、下図に示します。



※／は、制御線の本数を示しています。
　／7 は、7 本の制御線があるということです。

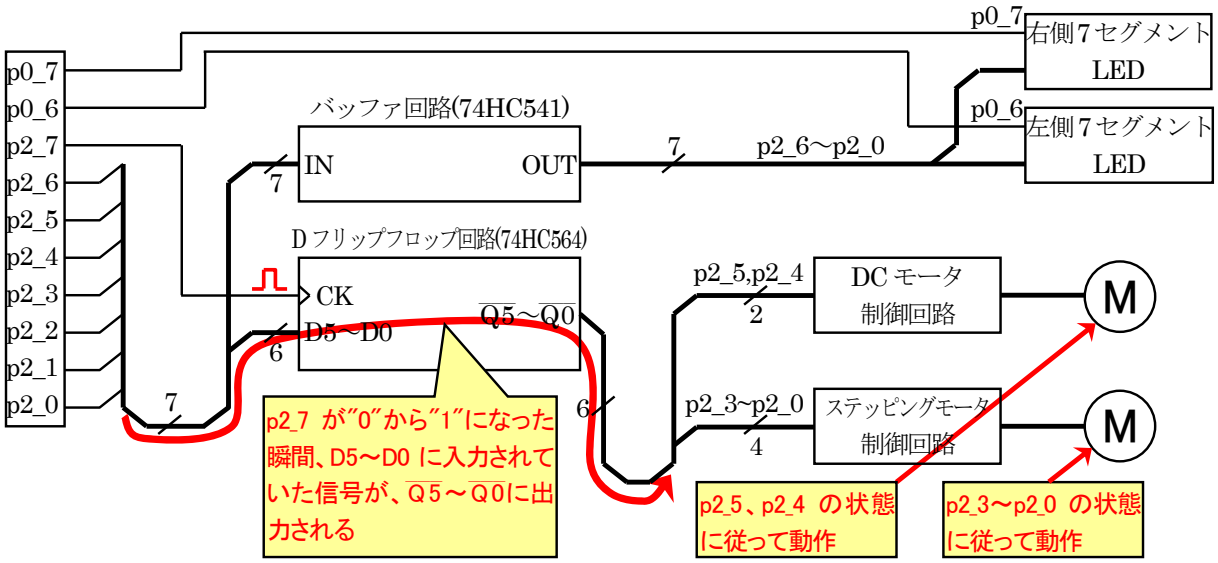
①7 セグメント LED の消灯

まず、p0_7 と p0_6 を″1″にして、7 セグメント LED を消灯させます。



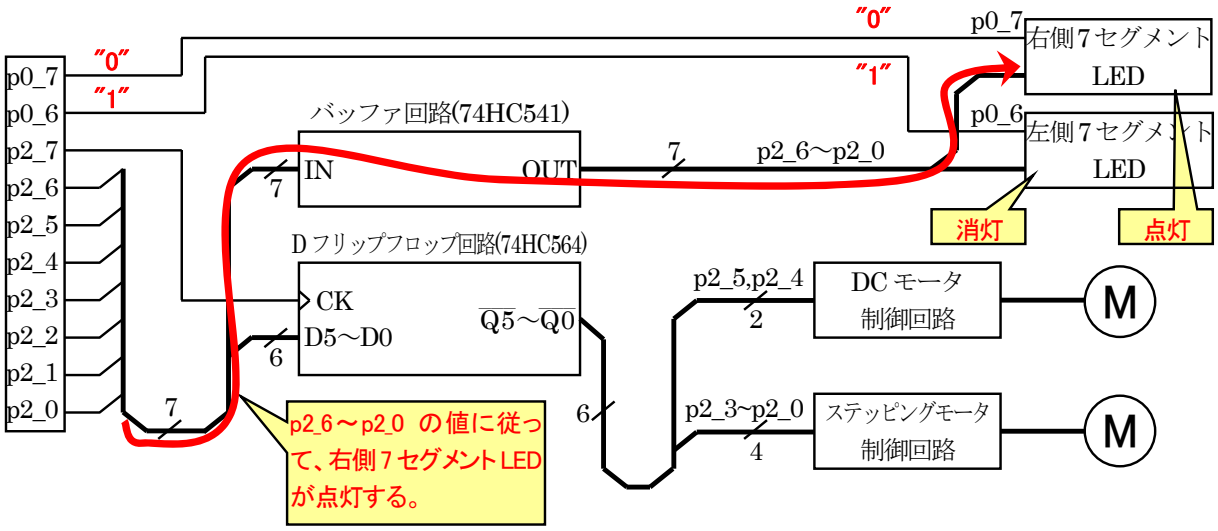
②ステッピングモータ、DC モータの制御

ステッピングモータを制御するために p2_3～p2_0、DC モータを制御するために p2_5、p2_4 の各端子からデータを出力します。まだ、各モータは前の動作のまま変化しません。p2_7 が"0"から"1"になった瞬間に 74HC564 の入力端子 D5～D0 の値が$\overline{Q5}$～$\overline{Q0}$から出力されます。よって、p2_7 を"0"→"1"にした瞬間に各モータの動作が変化します。次に p2_7 端子を"0"から"1"にするまで変化しません。



③7 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_6～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と DC 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。

4.7.4 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。1-2 相励磁で反時計回りに回転させるには「A → AとB → B → Bと$\overline{A}$ → $\overline{A}$ → $\overline{A}$と$\overline{B}$ → $\overline{B}$ → $\overline{B}$とA」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。

p0_0	p0_1	p0_2	p0_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
"1"	"1"	"0"	"0"	ON	ON	OFF	OFF
"0"	"1"	"0"	"0"	OFF	ON	OFF	OFF
"0"	"1"	"1"	"0"	OFF	ON	ON	OFF
"0"	"0"	"1"	"0"	OFF	OFF	ON	OFF
"0"	"0"	"1"	"1"	OFF	OFF	ON	ON
"0"	"0"	"0"	"1"	OFF	OFF	OFF	ON
"1"	"0"	"0"	"1"	ON	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→0x03→0x02→0x06→0x04→0x0c→0x08→0x09→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。

```
83 :  ////////////////////////////////////////////////////////////////
84 :  // ステッピングモータの励磁出力信号 1-2 相励磁
85 :  // bit3=D(/B) bit2=C(/A) bit1=B bit0=A
86 :  const signed char stm_data[] = {
87 :      0b00000001, 0b00000011, 0b00000010, 0b00000110,      // 0～3
88 :      0b00000100, 0b00001100, 0b00001000, 0b00001001,      // 4～7
89 :  };
```

添字([ ] の中に入れる数字のこと)に 0～7 の値をセットすると、0b00000001、0b00000011 ・・・ 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 ] & 0x7f;     // ポート 2(p2)には、0b00000011 の反転した値が設定される
```

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 7,6,5,4…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。今年の課題は回転させるスピードを変える問題が無いので、切り替える間隔は 50ms 固定とします。

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、3.75 度回転します(1-2 励磁のため)。1 回転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 3.75＝96

よって、96 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。

(4) 使用する変数

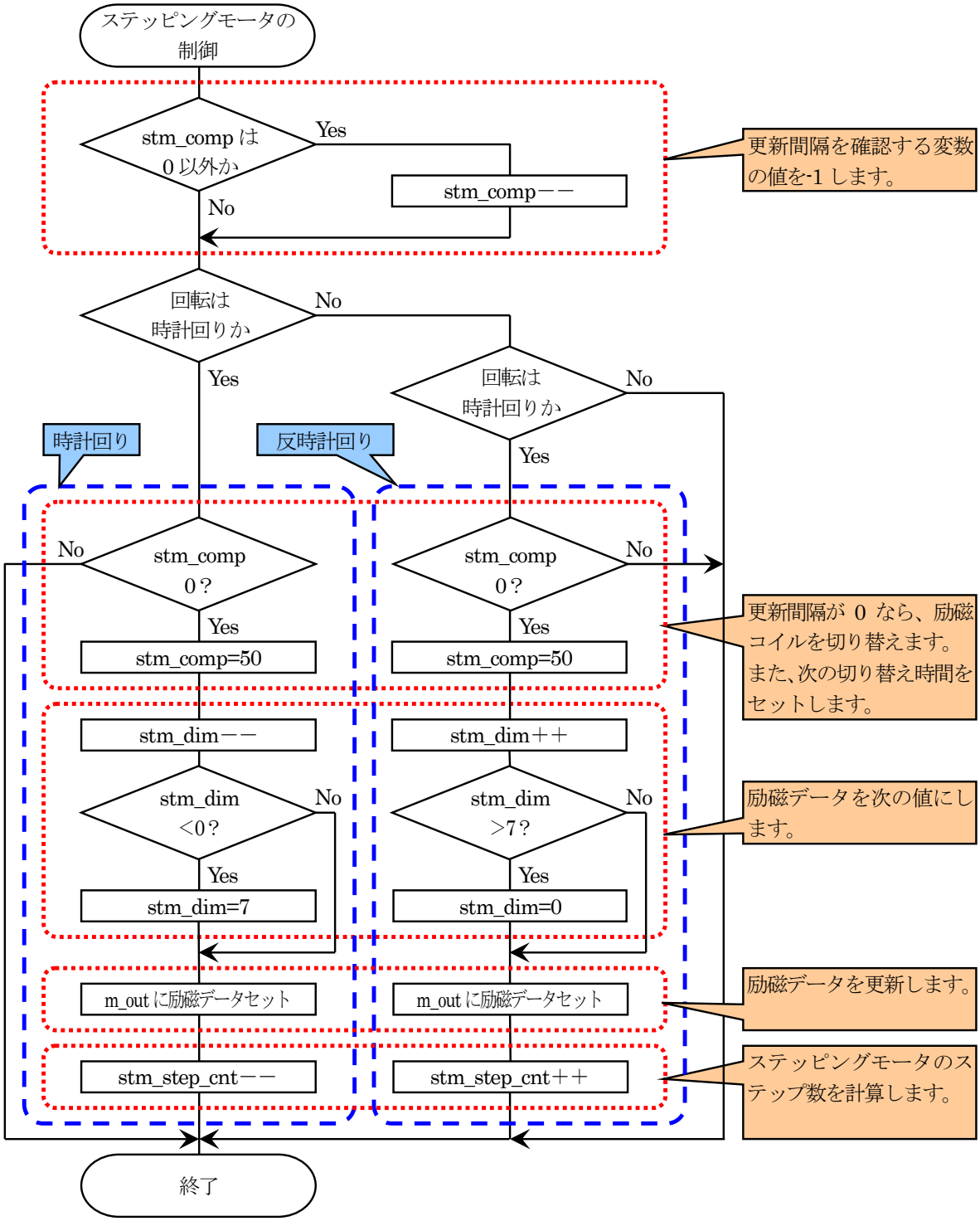
ステッピングモータ関連で使用する変数を、下記に示します。

変数	内容
stm_dir	ステッピングモータの回転方向を設定します。 1：時計周りに回転 0：停止 -1：反時計回りに回転 これらは、define 文で #define M_TOKEI　1　// 時計回りに回転 #define M_STOP　0　// 停止 #define M_HANTOKEI　-1　// 反時計回りに回転 と定義しています。使用例を下記に示します。 stm_dir = M_TOKEI　// 時計回りに回転 stm_dir = M_STOP　// 停止 stm_dir = M_HANTOKEI　// 反時計回りに回転
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。1ms ごとの割り込み内で、この変数を増減させます。 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。

stm_comp	励磁コイルを切り替える間隔を比較する変数です。main 関数では、この変数は操作しません。
	次のプログラムで 50ms ごとに励磁コイルを切り替えます(時計回りで制御の例)。
	
	if(stm_comp != 0) stm_comp--;　// 1ms ごとに-1 する
	if(stm_dir == M_TOKEI) {　// 時計回りか？
	　if(stm_comp == 0) {　// 励磁コイルを替える間隔のチェック
	　　// 次に励磁を切り替える間隔をセット
	　　stm_comp = 50;　// ms 単位でセット
	　　励磁コイルを切り替えるプログラム
	　}

stm_dim	stm_data 配列の添字です。main 関数では、この変数は操作しません。 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、0～7 の範囲なので、8 以上になったら 0 に、-1 以下になったら 7 にします。
m_out	ステッピングモータは p2_3～p2_0、DC モータは p2_5～p2_4 なので、ポート 2 にステッピングモータの励磁データを出力すると、DC モータのビットにも影響を与えてしまいます。そこで、m_out 変数を用意して、そこにステッピングモータの出力データと DC モータの出力データをセットして、その後、ポート 2 に二つのモータのデータを出力します。

(5) フローチャート



(6) プログラム

```
// グローバル変数の宣言

 83 :  ////////////////////////////////////////////////////////////
 84 :  // ステッピングモータの励磁出力信号 1-2 相励磁
 85 :  // bit3=D(/B) bit2=C(/A) bit1=B bit0=A
 86 :  const signed char stm_data[] = {
 87 :      0b00000001, 0b00000011, 0b00000010, 0b00000110,     // 0～3
 88 :      0b00000100, 0b00001100, 0b00001000, 0b00001001,     // 4～7
 89 :  };
 90 :
 91 :  // ステッピングモータ
 92 :  int     stm_dir;                        // 回転方向
 93 :  int     stm_step_cnt;                   // 動作したステップ数 96 で 1 回転
 94 :  int     stm_comp;                       // 励磁コイル切り替える間隔
 95 :  int     stm_dim;                        // 励磁出力信号の添字

// プログラム

176 :      // ステッピングモータの制御
177 :      if( stm_comp != 0 ) stm_comp--;     // 励磁ｺｲﾙを替える間隔を減らす
178 :
179 :      if( stm_dir == M_TOKEI ) {          // 時計回りか？
180 :          if( stm_comp == 0 ) {           // 励磁コイルを替える間隔のチェック
181 :              // 次に励磁を切り替える間隔をセット
182 :              stm_comp = 50;              // ms 単位でセット
183 :
184 :              // 励磁コイルの切り替え
185 :              stm_dim--;
186 :              if( stm_dim < 0 ) stm_dim = 7;
187 :
188 :              // 励磁データセット
189 :              m_out &= 0xf0;
190 :              m_out |= stm_data[ stm_dim ];
191 :
192 :              // ステップ数を増やす
193 :              stm_step_cnt++;
194 :          }
195 :      } else if( stm_dir == M_HANTOKEI ) { // 反時計回りか？
196 :          if( stm_comp == 0 ) {           // 励磁コイルを替える間隔のチェック
197 :              // 次に切り替える間隔をセット
198 :              stm_comp = 50;              // ms 単位でセット
199 :
200 :              // 励磁コイルの切り替え
201 :              stm_dim++;
202 :              if( stm_dim > 7 ) stm_dim = 0;
203 :
204 :              // 励磁データセット
205 :              m_out &= 0xf0;
206 :              m_out |= stm_data[ stm_dim ];
207 :
208 :              // ステップ数を減らす
209 :              stm_step_cnt--;
210 :          }
211 :      } else {
212 :          // 停止、前の出直値のまま何もせず
213 :      }
```

4.7.5 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。課題 3 では、低速回転、高速回転させなければいけませんが、今回は、main 関数で処理することとして、ここでは高速回転のみとします。そのため、単純に"0"または"1"を出力するだけで DC モータを制御可能です。

ポートと DC モータの動作の関係を、下表に示します。

p2_5	p2_4	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

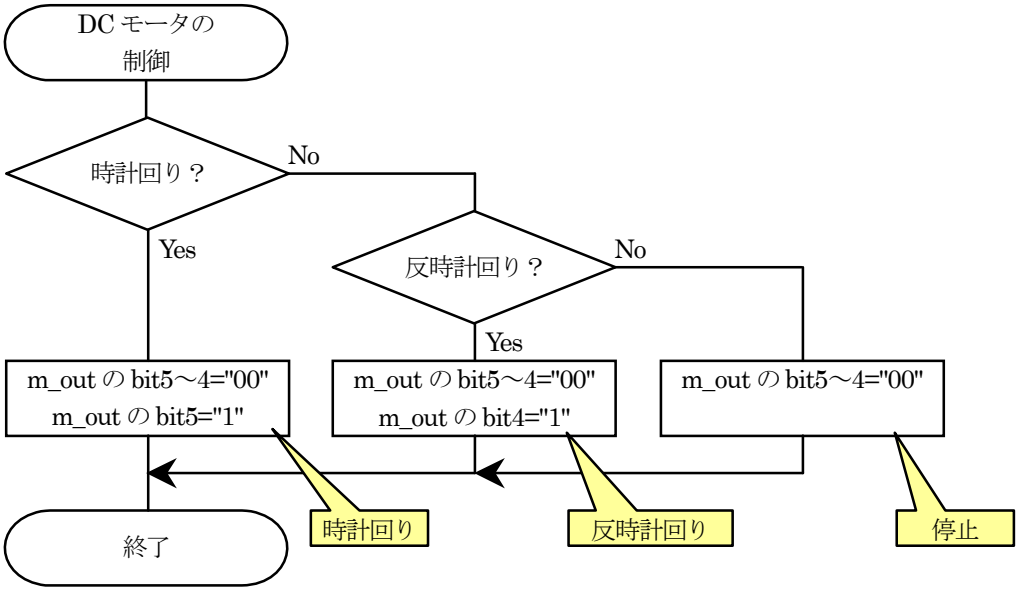
ストップは DC モータの端子間を解放、ブレーキは DC モータの端子間をショートさせます。今回モータを止めるのはストップを使います。

(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_dir	DC モータの回転方向を設定します。 1：時計周りに回転 0：停止 -1：反時計回りに回転 これらは、define 文で #define M_TOKEI　1　// 時計回りに回転 #define M_STOP　0　// 停止 #define M_HANTOKEI　-1　// 反時計回りに回転 と定義しています。使用例を下記に示します。 dcm_dir = M_TOKEI　// 時計回りに回転 dcm_dir = M_STOP　// 停止 dcm_dir = M_HANTOKEI　// 反時計回りに回転
m_out	ステッピングモータは p2_3～p2_0、DC モータは p2_5～p2_4 なので、ポート 2 に DC モータのデータを出力すると、ステッピングモータのビットにも影響を与えてしまいます。そこで、m_out 変数を用意して、そこにステッピングモータの出力データと DC モータの出力データをセットして、その後、ポート 2 に二つのモータのデータを出力します。

(3) フローチャート



(4) プログラム

```
// グローバル変数の宣言

 97 :  ////////////////////////////////////////////////////////////
 98 :  // DC モータ
 99 :  int     dcm_dir;                        // 回転方向

// プログラム

215 :      // DC モータの制御
216 :      if( dcm_dir == M_TOKEI ) {          // 時計回りか？
217 :          m_out &= 0xcf;
218 :          m_out |= 0x20;
219 :      } else if( dcm_dir == M_HANTOKEI ) { // 反時計回りか？
220 :          m_out &= 0xcf;
221 :          m_out |= 0x10;
222 :      } else {
223 :          // 停止
224 :          m_out &= 0xcf;
225 :      }
```

4.7.6 モータ出力処理

(1) モータへ信号を出力する方法

ステッピングモータとDC モータへ出力するデータを、m_out 変数に格納しました。m_out 変数の値をポート 2 へ出力して実際にモータを制御します。ただし、ポート 2 は左側 7 セグメント LED、右側 7 セグメント LED も接続されているので、混同しないようにしなければいけません。

(2) プログラム

```
227 :      // DC モータ、ステッピングモータへ出力
228 :      p0_7 = 1;                          // 7 セグメント LED 消灯(兼用のため)
229 :      p0_6 = 1;
230 :
231 :      p2 = ~m_out & 0x7f;                // DC ﾓｰﾀ、ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀへ信号出力
232 :
233 :      p2_7 = 1;                          // CK HIGH
234 :      p2_7 = 0;                          // CK LOW
```

行	詳細
228、229	p0_7 と p0_6 に"1"を出力して、左側 7 セグメント LED、右側 7 セグメント LED を消灯させます。
231	m_out 変数の値はポート 2 を通して 74HC564 の D5～D0 に入力されます。ただし、74HC564 は反転出力なので「~(チルダ)」を付けて出力値を反転させます。また、bit7 は 0 のままにするので、0x7f で AND 演算した値をポート 2 に出力します。 まだモータには出力されていません。
233	p2_7 を"1"にします。"1"にした瞬間、74HC564 の D5～D0 に入力されているポート 2 の出力データが、$\overline{Q5}$～$\overline{Q0}$端子から出力され、モータの回路へ信号が出力されます。
234	p2_7 を"0"にします。次に p2_7 を"1"にするまで 74HC564 の出力データは変化しません。よって、ポート 2 の値を変えてもモータの動作は変わりません。この間に 7 セグメント LED へデータを出力し表示させます。

4.7.7　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 a f b g e c d dp	dp 今回は未接続	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

上表のデータをポート 2 から出力し、p0_6 を"0"にすると左側 7 セグメント LED が点灯、p0_7 を"0"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。

```
55 :  // 7 セグメント LED の表示データ
56 :  const unsigned char seg_data[] = {
57 :      0b00111111,                    //  0 = 0
58 :      0b00000110,                    //  1 = 1
59 :      0b01011011,                    //  2 = 2
60 :      0b01001111,                    //  3 = 3
61 :      0b01100110,                    //  4 = 4
62 :      0b01101101,                    //  5 = 5
63 :      0b01111101,                    //  6 = 6
64 :      0b00100111,                    //  7 = 7
65 :      0b01111111,                    //  8 = 8
66 :      0b01101111,                    //  9 = 9
67 :      0b01110111,                    // 10 = A
68 :      0b01111100,                    // 11 = b
69 :      0b00111001,                    // 12 = C
70 :      0b01011110,                    // 13 = d
71 :      0b01111001,                    // 14 = E
72 :      0b01110001,                    // 15 = F
73 :      0b01010100,                    // 16 = OFF
74 :      0b00001000,                    // 17 = ON
75 :      0b01000000,                    // 18 = -
76 :  };
```

添字([ ] の中に入れる数字のこと)に 0～18 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 OFF、17 が ON、18 が"－"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 ] & 0x7f;     // ポート 2(p2)には、0b00001000 が設定されます。
```

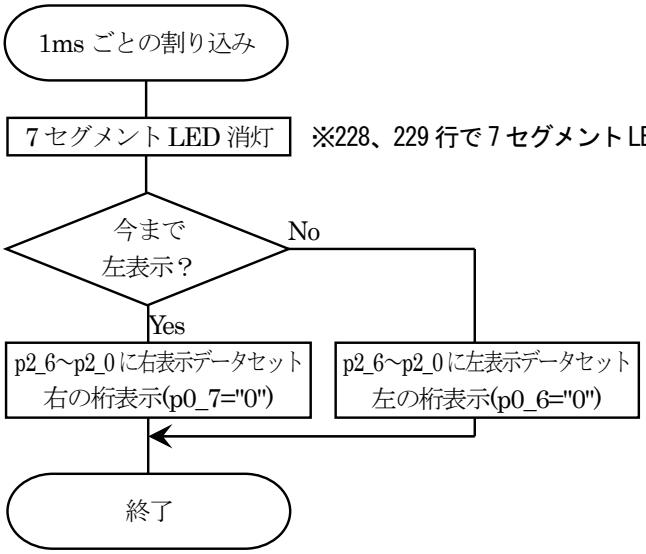
LED は"0"で点灯なので、反転して出力します。'A'を表示する値である 0b01110111 を反転すると、0b10001000 になります。また、bit7 は 74HC564 のクロック入力なので、0x7f で AND 演算を行い強制的に"0"にします。よって、ポート 2 には 0b000010000 がセットされます。このように、添字にそのまま 0～18 の値をセットすればポート 2 に出力する値が取り出せるので、いちいち 0 だから 0b00111111 をセットして･･･ 1 だから 0b00000110 をセットして･･･ というようにプログラムする必要が無くなります。

(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 ON、17 が OFF、18 が"－"をそれぞれ表示します。負の数なら消灯となります。 "ON"、"OFF"、"－"、消灯は、define 文で下記のように定義しています。 #define SEG_OFF　16　// 7 セグ OFF 表示 #define SEG_ON　17　// 7 セグ ON 表示 #define SEG_MAI　18　// 7 セグ マイナス表示 #define SEG_NULL　-1　// 7 セグ 消灯 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



(5) プログラム

```
// グローバル変数の宣言

  54 :  ////////////////////////////////////////////////////////////
  55 :  // 7 セグメント LED の表示データ
  56 :  const unsigned char seg_data[] = {
  57 :      0b00111111,                          //  0 = 0
  58 :      0b00000110,                          //  1 = 1
  59 :      0b01011011,                          //  2 = 2
  60 :      0b01001111,                          //  3 = 3
  61 :      0b01100110,                          //  4 = 4
  62 :      0b01101101,                          //  5 = 5
  63 :      0b01111101,                          //  6 = 6
  64 :      0b00100111,                          //  7 = 7
  65 :      0b01111111,                          //  8 = 8
  66 :      0b01101111,                          //  9 = 9
  67 :      0b01110111,                          // 10 = A
  68 :      0b01111100,                          // 11 = b
  69 :      0b00111001,                          // 12 = C
  70 :      0b01011110,                          // 13 = d
  71 :      0b01111001,                          // 14 = E
  72 :      0b01110001,                          // 15 = F
  73 :      0b01010100,                          // 16 = OFF
  74 :      0b00001000,                          // 17 = ON
  75 :      0b01000000,                          // 18 = -
  76 :  };
  77 :
  78 :  // 7 セグメント LED の表示値
  79 :  int     seg_left  = SEG_NULL;            // 左の桁　表示値
  80 :  int     seg_right = SEG_NULL;            // 右の桁　表示値
  81 :  int     seg_digit;                       // 0:左表示 1:右表示

// プログラム

 236 :      // 7 セグメント LED の表示　左と右の桁を交互に表示
 237 :      if( seg_digit == 0 ) {               // 左表示中なら
 238 :          // 右を表示
 239 :          seg_digit = 1;
 240 :          if( seg_right >= 0 ) {
 241 :              p2   = ~seg_data[seg_right] & 0x7f; // 値のセット
 242 :              p0_7 = 0;                           // 右 ON
 243 :          }
 244 :      } else {
 245 :          // 左を表示
 246 :          seg_digit = 0;
 247 :          if( seg_left >= 0 ) {
 248 :              p2   = ~seg_data[seg_left] & 0x7f;  // 値のセット
 249 :              p0_6 = 0;                           // 左 ON
 250 :          }
 251 :      }
```

行	詳細
237	seg_digit 変数が 0 なら左側 7 セグメント LED 表示中です。 seg_digit 変数が 0 かチェックして、0 なら 239～243 行を実行して右側 7 セグメント LED を表示、それ以外(右側表示中)なら、else 文の 246～250 行を実行します。
239	seg_digit 変数を 1 にして、右側表示にします。
240	表示値がマイナスなら、何もセットせずに終了します。右側 7 セグメント LED は消灯です。
241	seg_data 配列から表示データを読み込んで、ポート 2 に設定します。
242	p0_7 を"0"にして、右側 7 セグメント LED を点灯します。
246～250	左側 7 セグメント LED を表示させる処理です。内容は、右側と同様です。

4.8 「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、入力信号を取り込むプログラム、出力回路へ信号を出力するプログラムなどを「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 ： // 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　共通ファイル
 3 ： // 座席番号：●
 4 ： // 氏　　名：○○　○○
 5 ： //
 6 ： // Copyright (C) 2011 ルネサスマイコンカーラリー事務局
 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 ： #include "r8c38a_lib.h"                 // R8C/38A マイコンライブラリ
14 ：
15 ： ////////////////////////////////////////////////////////////////////////
16 ： // シンボル定義
17 ： ////////////////////////////////////////////////////////////////////////
18 ： // seg_left と seg_right 変数に入れる値
19 ： #define SEG_OFF     16                 // 7 セグ OFF 表示
20 ： #define SEG_ON      17                 // 7 セグ ON 表示
21 ： #define SEG_MAI     18                 // 7 セグ マイナス
22 ： #define SEG_NULL    -1                 // 7 セグ 消灯
23 ：
24 ： // モータ(ステッピングモータ、DC モータ共通)
25 ： #define M_TOKEI     1                  // 時計回りに回転
26 ： #define M_STOP      0                  // 停止
27 ： #define M_HANTOKEI  -1                 // 反時計回りに回転
28 ：
29 ： ////////////////////////////////////////////////////////////////////////
30 ： // グローバル変数の宣言
31 ： ////////////////////////////////////////////////////////////////////////
32 ：
33 ： ////////////////////////////////////////////////////////////
34 ： // カウンタ
35 ： long    cnt0;                           // 1ms ごとに+1
36 ： int     cnt0_flag;                      // 0:cnt0 のカウントを停止
37 ：
38 ： int     cnt10ms;                        // 10ms ごとの処理用
39 ：
40 ： ////////////////////////////////////////////////////////////
41 ： // フォトインタラプタの状態
42 ： int     phsw;                           // 0:透過 1:遮断
43 ： int     phsw_syadanflag;                // 透過→遮断された瞬間に 1
44 ：
45 ： ////////////////////////////////////////////////////////////
46 ： // タクトスイッチの状態
47 ： int     tcsw;                           // 0:OFF 1:ON
48 ： int     tcsw_offflag;                   // ON→OFF になった瞬間に 1
49 ：
50 ： ////////////////////////////////////////////////////////////
51 ： // トグルスイッチの状態
52 ： int     tgsw;                           // 0:OFF 1:ON
53 ：
```

```
 54 :  ////////////////////////////////////////////////////////////
 55 :  // 7 セグメント LED の表示データ
 56 :  const unsigned char seg_data[] = {
 57 :      0b00111111,                        //  0 = 0
 58 :      0b00000110,                        //  1 = 1
 59 :      0b01011011,                        //  2 = 2
 60 :      0b01001111,                        //  3 = 3
 61 :      0b01100110,                        //  4 = 4
 62 :      0b01101101,                        //  5 = 5
 63 :      0b01111101,                        //  6 = 6
 64 :      0b00100111,                        //  7 = 7
 65 :      0b01111111,                        //  8 = 8
 66 :      0b01101111,                        //  9 = 9
 67 :      0b01110111,                        // 10 = A
 68 :      0b01111100,                        // 11 = b
 69 :      0b00111001,                        // 12 = C
 70 :      0b01011110,                        // 13 = d
 71 :      0b01111001,                        // 14 = E
 72 :      0b01110001,                        // 15 = F
 73 :      0b01010100,                        // 16 = OFF
 74 :      0b00001000,                        // 17 = ON
 75 :      0b01000000,                        // 18 = -
 76 :  };
 77 :
 78 :  // 7 セグメント LED の表示値
 79 :  int     seg_left  = SEG_NULL;          // 左の桁　表示値
 80 :  int     seg_right = SEG_NULL;          // 右の桁　表示値
 81 :  int     seg_digit;                     // 0:左表示 1:右表示
 82 :
 83 :  ////////////////////////////////////////////////////////////
 84 :  // ステッピングモータの励磁出力信号 1-2 相励磁
 85 :  // bit3=D(/B) bit2=C(/A) bit1=B bit0=A
 86 :  const signed char stm_data[] = {
 87 :      0b00000001, 0b00000011, 0b00000010, 0b00000110,     // 0～3
 88 :      0b00000100, 0b00001100, 0b00001000, 0b00001001,     // 4～7
 89 :  };
 90 :
 91 :  // ステッピングモータ
 92 :  int     stm_dir;                       // 回転方向
 93 :  int     stm_step_cnt;                  // 動作したステップ数 96 で 1 回転
 94 :  int     stm_comp;                      // 励磁コイル切り替える間隔
 95 :  int     stm_dim;                       // 励磁出力信号の添字
 96 :
 97 :  ////////////////////////////////////////////////////////////
 98 :  // DC モータ
 99 :  int     dcm_dir;                       // 回転方向
100 :
101 :  ////////////////////////////////////////////////////////////
102 :  // DC モータ、ステッピングモータのポート出力データ　バッファ
103 :  unsigned char m_out = 0b00000001;       //  初期値は stm_data[0]の値
104 :
105 :  ////////////////////////////////////////////////////////////////////////
106 :  // R8C/38A スペシャルファンクションレジスタ(SFR)の初期化
107 :  ////////////////////////////////////////////////////////////////////////
108 :  void init( void )
109 :  {
110 :      // CPU の動作クロックを XIN クロックにする
111 :      init_xin_clk();
112 :
113 :      // タイマ RB で 1ms ごとに割り込みを発生
114 :      set_timer_b( INTERVAL_INT, 1000 );
115 :
116 :      // ポートの入出力設定
117 :      pd( 0 ,0xf8 );                     // 7-6:回路③ 2-0:回路①
118 :      pd( 1, 0xf0 );                     // 5:RXD0 4:TXD0 3-0:DIP SW
119 :      pd( 2 ,0xff );                     // 7-0:回路③
120 :      pd( 3 ,0xff );
121 :      pd( 4 ,0xb8 );                     // 7:XOUT 6:XIN 5:LED 2:VREF
122 :      pd( 5 ,0xff );
123 :      pd( 6 ,0xff );
124 :      pd( 7 ,0xff );
```

```
125 :     pd( 8 ,0xff );
126 :     pd( 9 ,0xff );
127 :
128 :     // 割り込み許可
129 :     ei();
130 : }
131 :
132 : ////////////////////////////////////////////////////////////////////////
133 : // タイマ RB 1ms ごとの割り込み処理
134 : ////////////////////////////////////////////////////////////////////////
135 : #pragma interrupt intTRB( vect = 24 )
136 : void intTRB( void )
137 : {
138 :     // カウンタカウントアップ
139 :     if( cnt0_flag != 0 ) {
140 :         cnt0++;
141 :     }
142 :
143 :     // フォトインタラプタ、スイッチは 10ms ごとにチェックする
144 :     cnt10ms++;
145 :     if( cnt10ms >= 10 ) {
146 :         cnt10ms = 0;
147 :
148 :         // フォトインタラプタの値取り込み
149 :         if( p0_0 == 1 ) {               // 遮断なら
150 :             if( phsw == 0 ) {
151 :                 phsw_syadanflag = 1;    // 遮断した瞬間なら 1
152 :             }
153 :             phsw = 1;
154 :         } else {                        // 透過なら
155 :             phsw = 0;
156 :         }
157 :
158 :         // タクトスイッチの値取り込み
159 :         if( p0_1 == 0 ) {               // OFF なら
160 :             if( tcsw == 1 ) {
161 :                 tcsw_offflag = 1;       // OFF になった瞬間なら 1
162 :             }
163 :             tcsw = 0;
164 :         } else {                        // ON なら
165 :             tcsw = 1;
166 :         }
167 :
168 :         // トグルスイッチの値取り込み
169 :         if( p0_2 == 0 ) {               // OFF なら
170 :             tgsw = 0;
171 :         } else {                        // ON なら
172 :             tgsw = 1;
173 :         }
174 :     }
175 :
176 :     // ステッピングモータの制御
177 :     if( stm_comp != 0 ) stm_comp--;     // 励磁ｺｲﾙを替える間隔を減らす
178 :
179 :     if( stm_dir == M_TOKEI ) {          // 時計回りか？
180 :         if( stm_comp == 0 ) {           // 励磁コイルを替える間隔のチェック
181 :             // 次に励磁を切り替える間隔をセット
182 :             stm_comp = 50;              // ms 単位でセット
183 :
184 :             // 励磁コイルの切り替え
185 :             stm_dim--;
186 :             if( stm_dim < 0 ) stm_dim = 7;
187 :
188 :             // 励磁データセット
189 :             m_out &= 0xf0;
190 :             m_out |= stm_data[ stm_dim ];
191 :
192 :             // ステップ数を増やす
193 :             stm_step_cnt++;
194 :         }
```

```
195 :     } else if( stm_dir == M_HANTOKEI ) { // 反時計回りか？
196 :         if( stm_comp == 0 ) {           // 励磁コイルを替える間隔のチェック
197 :             // 次に切り替える間隔をセット
198 :             stm_comp = 50;              // ms 単位でセット
199 :
200 :             // 励磁コイルの切り替え
201 :             stm_dim++;
202 :             if( stm_dim > 7 ) stm_dim = 0;
203 :
204 :             // 励磁データセット
205 :             m_out &= 0xf0;
206 :             m_out |= stm_data[ stm_dim ];
207 :
208 :             // ステップ数を減らす
209 :             stm_step_cnt--;
210 :         }
211 :     } else {
212 :         // 停止、前の出直値のまま何もせず
213 :     }
214 :
215 :     // DC モータの制御
216 :     if( dcm_dir == M_TOKEI ) {          // 時計回りか？
217 :         m_out &= 0xcf;
218 :         m_out |= 0x20;
219 :     } else if( dcm_dir == M_HANTOKEI ) { // 反時計回りか？
220 :         m_out &= 0xcf;
221 :         m_out |= 0x10;
222 :     } else {
223 :         // 停止
224 :         m_out &= 0xcf;
225 :     }
226 :
227 :     // DC モータ、ステッピングモータへ出力
228 :     p0_7 = 1;                           // 7 セグメント LED 消灯(兼用のため)
229 :     p0_6 = 1;
230 :
231 :     p2 = ~m_out & 0x7f;                 // DC ﾓｰﾀ、ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀへ信号出力
232 :
233 :     p2_7 = 1;                           // CK HIGH
234 :     p2_7 = 0;                           // CK LOW
235 :
236 :     // 7 セグメント LED の表示　左と右の桁を交互に表示
237 :     if( seg_digit == 0 ) {              // 左表示中なら
238 :         // 右を表示
239 :         seg_digit = 1;
240 :         if( seg_right >= 0 ) {
241 :             p2   = ~seg_data[seg_right] & 0x7f; // 値のセット
242 :             p0_7 = 0;                           // 右 ON
243 :         }
244 :     } else {
245 :         // 左を表示
246 :         seg_digit = 0;
247 :         if( seg_left >= 0 ) {
248 :             p2   = ~seg_data[seg_left] & 0x7f;  // 値のセット
249 :             p0_6 = 0;                           // 左 ON
250 :         }
251 :     }
252 : }
253 :
254 : ////////////////////////////////////////////////////////////////////////////
255 : // End of File
256 : ////////////////////////////////////////////////////////////////////////////
```

4.9　課題1

4.9.1　課題

(1) タクトスイッチの「ON」、「OFF」の状態を 7 セグメント LED に表示する。

タクトスイッチが「OFF」　　タクトスイッチが「ON」

タクトスイッチの状態を示す 7 セグメント LED の表示

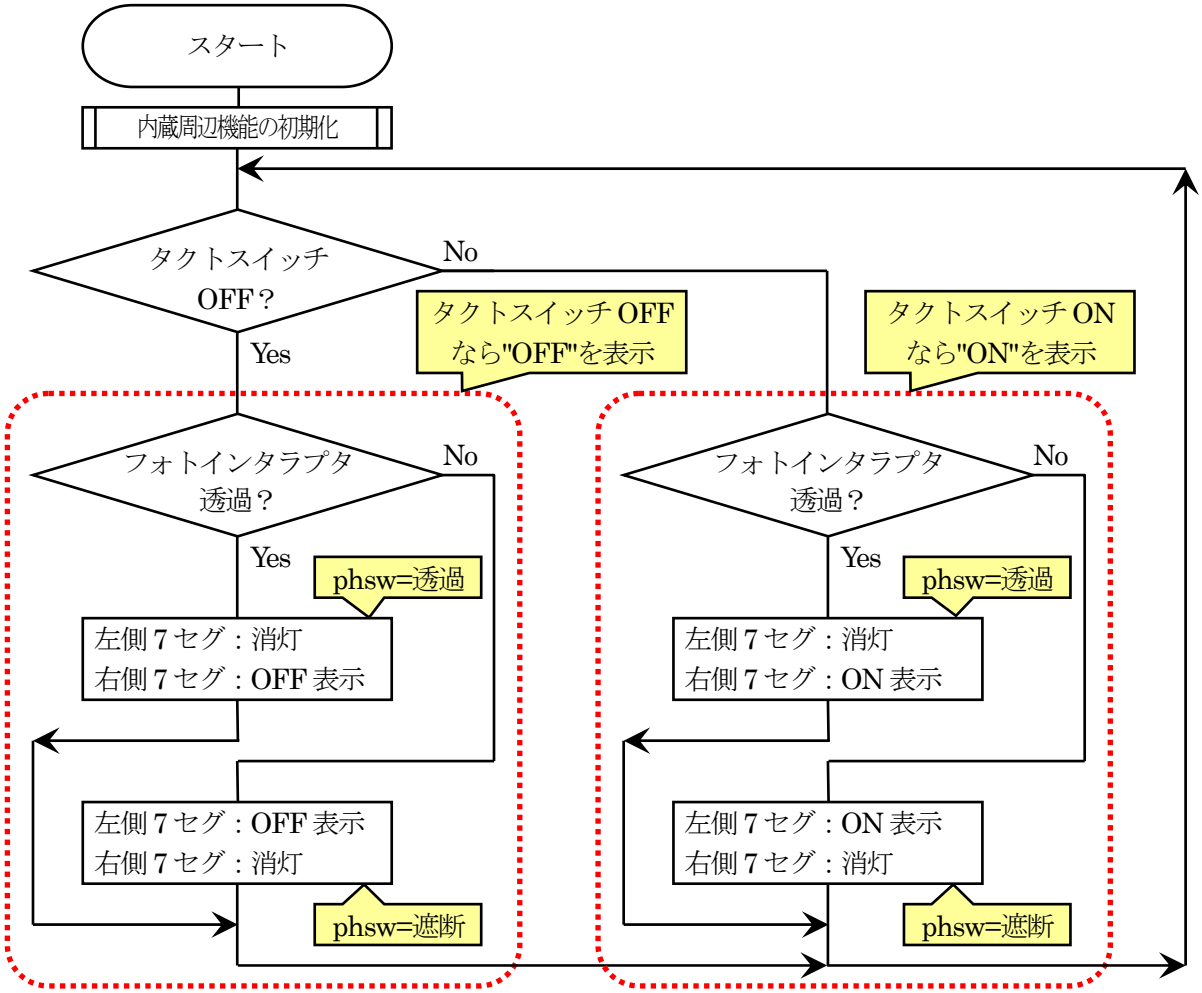
(2) 表示する 7 セグメント LED の位置は、フォトインタラプタの状態で次のように決まる。

(ア)　フォトインタラプタが「透過」で、右側の 7 セグメント LED に表示する。

(イ)　フォトインタラプタが「遮断」で、左側の 7 セグメント LED に表示する。

▲大会当日配付資料より抜粋

4.9.2　フローチャート



4.9.3 プログラム例

```
 1 : ////////////////////////////////////////////////////////////////////////
 2 : // 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　課題 1
 3 : // 座席番号：●
 4 : // 氏　　名：○○　○○
 5 : //
 6 : // Copyright (C) 2011 ルネサスマイコンカーラリー事務局
 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 ) {
22 :         if( tcsw == 0 ) {
23 :             // タクトスイッチ OFF なら
24 :             if( phsw == 0 ) {         // フォトインタラプタが透過なら
25 :                 seg_left  = SEG_NULL;
26 :                 seg_right = SEG_OFF;
27 :             } else {                  // フォトインタラプタが遮断なら
28 :                 seg_left  = SEG_OFF;
29 :                 seg_right = SEG_NULL;
30 :             }
31 :         } else {
32 :             // タクトスイッチ ON なら
33 :             if( phsw == 0 ) {         // フォトインタラプタが透過なら
34 :                 seg_left  = SEG_NULL;
35 :                 seg_right = SEG_ON;
36 :             } else {                  // フォトインタラプタが遮断なら
37 :                 seg_left  = SEG_ON;
38 :                 seg_right = SEG_NULL;
39 :             }
40 :         }
41 :     }
42 : }
43 :
44 : ////////////////////////////////////////////////////////////////////////
45 : // end of file
46 : ////////////////////////////////////////////////////////////////////////.
```

4.9.4 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。
22	タクトスイッチが OFF かどうかチェックします。OFF なら 23～30 行、ON なら 32～39 行を実行します。
23～30	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。
32～39	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。

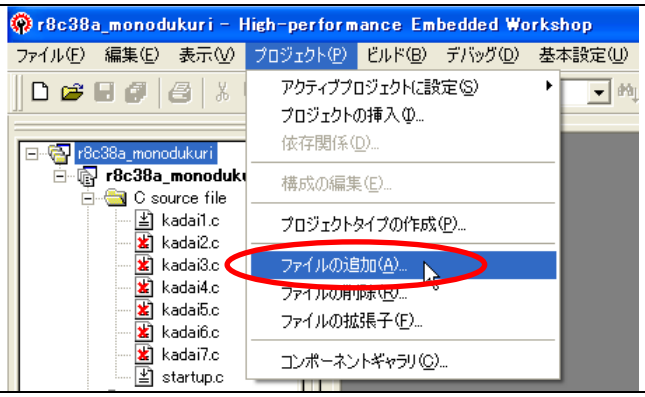
4.9.5 割り込み、共通ファイルを使わないプログラム例「kadai1_kanni.c」

割り込み、共通ファイル(common.c)を使わないプログラム例を下記に示します。動作は、使用したときのフローチャートと同様です。

```
 1 : ////////////////////////////////////////////////////////////////////////
 2 : // 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　課題 1
 3 : //                                        （共通ファイル、割り込み　未使用版）
 4 : // 座席番号：●
 5 : // 氏　　名：○○　○○
 6 : //
 7 : // Copyright (C) 2011 ルネサスマイコンカーラリー事務局
 8 : ////////////////////////////////////////////////////////////////////////
 9 :
10 : ////////////////////////////////////////////////////////////////////////
11 : // インクルード
12 : ////////////////////////////////////////////////////////////////////////
13 : #include "sfr_r838a.h"                  // R8C/38A SFR の定義ファイル
14 : #include "r8c38a_lib.h"                 // R8C/38A マイコンライブラリ
15 :
16 : ////////////////////////////////////////////////////////////////////////
17 : // シンボル定義
18 : ////////////////////////////////////////////////////////////////////////
19 : #define PHSW            p0_0            // フォトインタラプタ
20 : #define TCSW            p0_1            // タクトスイッチ
21 : #define SEG_LEFT        1               // 左側 7 セグメント LED を選択
22 : #define SEG_RIGHT       2               // 右側 7 セグメント LED を選択
23 : #define OFF             0b01010100      // 7 セグメント LED 表示データ：OFF
24 : #define ON              0b00001000      // 7 セグメント LED 表示データ：ON
25 : #define NULL            0b00000000      // 7 セグメント LED 表示データ：非表示
26 :
27 : ////////////////////////////////////////////////////////////////////////
28 : // R8C/38A スペシャルファンクションレジスタ(SFR)の初期化
29 : ////////////////////////////////////////////////////////////////////////
30 : void init( void )
31 : {
32 :     // CPU の動作クロックを XIN クロックにする
33 :     init_xin_clk();
34 :
35 :     // ポートの入出力設定
36 :     pd( 0 ,0xf8 );                     // 7-6:回路③ 2-0:回路①
37 :     pd( 1, 0xf0 );                     // 5:RXD0 4:TXD0 3-0:DIP SW
38 :     pd( 2 ,0xff );                     // 7-0:回路③
39 :     pd( 3 ,0xff );
40 :     pd( 4 ,0xb8 );                     // 7:XOUT 6:XIN 5:LED 2:VREF
41 :     pd( 5 ,0xff );
42 :     pd( 6 ,0xff );
43 :     pd( 7 ,0xff );
44 :     pd( 8 ,0xff );
45 :     pd( 9 ,0xff );
46 : }
47 :
48 : ////////////////////////////////////////////////////////////////////////
49 : // 7 セグメント LED の制御
50 : ////////////////////////////////////////////////////////////////////////
51 : void seg_out( int keta, unsigned char data )
52 : {
53 :     p0_6 = 1;                          // 左側 7 セグ 消灯
54 :     p0_7 = 1;                          // 右側 7 セグ 消灯
55 :
56 :     // 7 セグメント LED にデータを出力
57 :     p2 = ~data & 0x7f;
58 :
59 :     // 左側または右側表示
60 :     if( keta == SEG_LEFT ) {
61 :         p0_6 = 0;                      // 左側 7 セグ 点灯
62 :     } else if( keta == SEG_RIGHT ) {
63 :         p0_7 = 0;                      // 右側 7 セグ 点灯
64 :     }
```

```
 65 :
 66 :     // 点灯時間
 67 :     timer_ms( 2 );
 68 : }
 69 :
 70 : ////////////////////////////////////////////////////////////////////////
 71 : // メイン関数
 72 : ////////////////////////////////////////////////////////////////////////
 73 : void main( void )
 74 : {
 75 :     init();                          // 内蔵周辺機能の初期化
 76 :
 77 :     while( 1 ) {
 78 :         if( TCSW == 0 ) {
 79 :             // タクトスイッチ OFF なら
 80 :             if( PHSW == 0 ) {        // フォトインタラプタが透過なら
 81 :                 seg_out( SEG_LEFT , NULL );
 82 :                 seg_out( SEG_RIGHT, OFF  );
 83 :             } else {                 // フォトインタラプタが遮断なら
 84 :                 seg_out( SEG_LEFT , OFF  );
 85 :                 seg_out( SEG_RIGHT, NULL );
 86 :             }
 87 :         } else {
 88 :             // タクトスイッチ ON なら
 89 :             if( PHSW == 0 ) {        // フォトインタラプタが透過なら
 90 :                 seg_out( SEG_LEFT , NULL );
 91 :                 seg_out( SEG_RIGHT, ON   );
 92 :             } else {                 // フォトインタラプタが遮断なら
 93 :                 seg_out( SEG_LEFT , ON   );
 94 :                 seg_out( SEG_RIGHT, NULL );
 95 :             }
 96 :         }
 97 :     }
 98 : }
 99 :
100 : ////////////////////////////////////////////////////////////////////////
101 : // end of file
102 : ////////////////////////////////////////////////////////////////////////
```

プログラムの登録方法を下記に示します。

1		「プロジェクト→ファイルの追加」を選択します。
2		「kadai1_kanni.c」を選択、追加をクリックします。 「kadai1_kanni.c」以外の課題ファイル（kadai1.c など）を右クリックで「ビルドから除外」します。 「ビルド→ビルド」で MOT ファイルが作成されます。

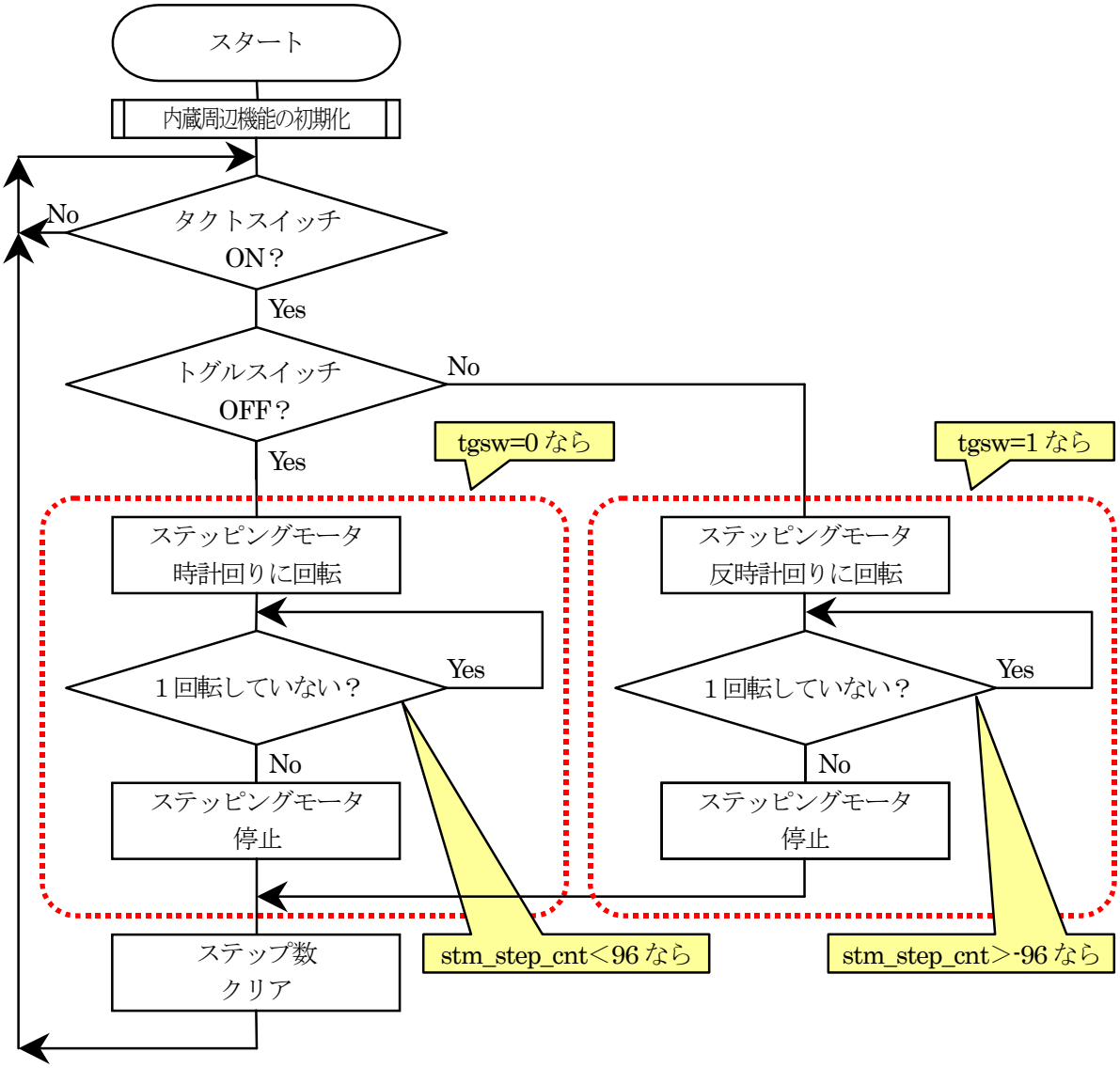
4.10　課題 2

4.10.1　課題

(1) プログラムを開始した後、ステッピングモータの目盛盤を手で動かし 0 の位置を指示針に合わせる。
(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を開始し一回転して停止する。
(3) 回転方向は、タクトスイッチを「ON」する直前のトグルスイッチの状態により決まる。
　(ア)　トグルスイッチが「OFF」で、時計回りに回転する。
　(イ)　トグルスイッチが「ON」で、反時計回りに回転する。
(4) ステッピングモータが一回転し停止した後、(2)以降の動作が行える。

▲大会当日配付資料より抜粋

4.10.2　フローチャート



4.10.3 プログラム

```
 1 : ////////////////////////////////////////////////////////////////////////
 2 : // 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　課題２
 3 : // 座席番号：●
 4 : // 氏　　名：○○　○○
 5 : //
 6 : // Copyright (C) 2011 ルネサスマイコンカーラリー事務局
 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 ) {
22 :         if( tcsw == 1 ) {             // タクトスイッチ ON なら
23 :             if( tgsw == 0 ) {         // トグルスイッチ OFF なら
24 :                 // ステッピングモータを時計回りに回す
25 :                 stm_dir = M_TOKEI;    // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ：時計回り
26 :
27 :                 // 1 回転するまで待つ(1 回転 96 ステップ)
28 :                 while( stm_step_cnt < 96 );
29 :
30 :                 // 停止
31 :                 stm_dir = M_STOP;     // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ：停止
32 :             } else {                  // トグルスイッチ ON なら
33 :                 // ステッピングモータを反時計回りに回す
34 :                 stm_dir = M_HANTOKEI; // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ：反時計回り
35 :
36 :                 // 1 回転するまで待つ(1 回転 96 ステップ)
37 :                 while( stm_step_cnt > -96 );
38 :
39 :                 // 停止
40 :                 stm_dir = M_STOP;     // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ：停止
41 :             }
42 :             stm_step_cnt = 0;         // ステップ数をクリア
43 :         }
44 :     }
45 : }
46 :
47 : ////////////////////////////////////////////////////////////////////////
48 : // end of file
49 : ////////////////////////////////////////////////////////////////////////
```

4.10.4 プログラムの解説

行	詳細
25～31	25 行でステッピングモータを時計回りに回転するよう設定します。このときステッピングモータが 1 ステップ進むたびに stm_step_cnt 変数も増えていきます。1 回転 96 ステップなので 96 回以下なら 1 回転していないと判断して回し続けます。96 になったら 1 回転したと判断して 31 行で停止させます。
34～40	34 行でステッピングモータを反時計回りに回転するよう設定します。このときステッピングモータが 1 ステップ進むたびに stm_step_cnt 変数も**減っていきます**。1 回転 96 ステップなので**-96 回より大きいなら** 1 回転していないと判断して回し続けます。-96 になったら 1 回転したと判断して 40 行で停止させます。
42	再度実行するときに備えて、stm_step_cnt を 0 にしておきます。ちょうど 1 回転しているので目盛りは 0 です。この変数を 0 にしても問題ありません。

4.10.5 割り込み、共通ファイルを使わないプログラム例「kadai2_kanni.c」

割り込み、共通ファイル(common.c)を使わないプログラム例を下記に示します。動作は、使用したときのフローチャートと同様です。

```
 1 :  ////////////////////////////////////////////////////////////////////////
 2 :  // 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　課題２
 3 :  //                                          （共通ファイル、割り込み　未使用版）
 4 :  // 座席番号：●
 5 :  // 氏　　名：○○　○○
 6 :  //
 7 :  // Copyright (C) 2011 ルネサスマイコンカーラリー事務局
 8 :  ////////////////////////////////////////////////////////////////////////
 9 :
10 :  ////////////////////////////////////////////////////////////////////////
11 :  // インクルード
12 :  ////////////////////////////////////////////////////////////////////////
13 :  #include "sfr_r838a.h"                  // R8C/38A SFR の定義ファイル
14 :  #include "r8c38a_lib.h"                 // R8C/38A マイコンライブラリ
15 :
16 :  ////////////////////////////////////////////////////////////////////////
17 :  // シンボル定義
18 :  ////////////////////////////////////////////////////////////////////////
19 :  #define TCSW            p0_1            // タクトスイッチの端子
20 :  #define TGSW            p0_2            // トグルスイッチの端子
21 :
22 :  #define STM_STOP        0x0f            // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ:ｽﾄｯﾌﾟ
23 :  #define DCM_STOP        0x00            // DC ﾓｰﾀ:ｽﾄｯﾌﾟ
24 :
25 :  ////////////////////////////////////////////////////////////
26 :  // ステッピングモータの励磁出力信号 1-2 相励磁
27 :  // bit3=D(/B) bit2=C(/A) bit1=B bit0=A  0:ON 1:OFF
28 :  const signed char stm_data[] = {
29 :      0b00001110, 0b00001100, 0b00001101, 0b00001001,     // 0～3
30 :      0b00001011, 0b00000011, 0b00000111, 0b00000110,     // 4～7
31 :  };
32 :
33 :  ////////////////////////////////////////////////////////////////////////
34 :  // R8C/38A スペシャルファンクションレジスタ(SFR)の初期化
35 :  ////////////////////////////////////////////////////////////////////////
36 :  void init( void )
37 :  {
38 :      // CPU の動作クロックを XIN クロックにする
39 :      init_xin_clk();
40 :
41 :      // ポートの入出力設定
42 :      pd( 0 ,0xf8 );                    // 7-6:回路③ 2-0:回路①
43 :      pd( 1, 0xf0 );                    // 5:RXD0 4:TXD0 3-0:DIP SW
44 :      pd( 2 ,0xff );                    // 7-0:回路③
45 :      pd( 3 ,0xff );
46 :      pd( 4 ,0xb8 );                    // 7:XOUT 6:XIN 5:LED 2:VREF
47 :      pd( 5 ,0xff );
48 :      pd( 6 ,0xff );
49 :      pd( 7 ,0xff );
50 :      pd( 8 ,0xff );
51 :      pd( 9 ,0xff );
52 :  }
53 :
```

```
 54 : ////////////////////////////////////////////////////////////////////////
 55 : // ステッピングモータと DC モータの制御
 56 : ////////////////////////////////////////////////////////////////////////
 57 : void motor( unsigned char data )
 58 : {
 59 :     p0_6 = 1;                          // 左側 7 セグ 消灯
 60 :     p0_7 = 1;                          // 右側 7 セグ 消灯
 61 :
 62 :     p2   = data;                       // 74HC564 の入力端子に出力
 63 :     p2_7 = 1;                          // 74HC564 の CK 立ち上げ→モータ出力
 64 :     p2_7 = 0;
 65 : }
 66 :
 67 : ////////////////////////////////////////////////////////////////////////
 68 : // メイン関数
 69 : ////////////////////////////////////////////////////////////////////////
 70 : void main( void )
 71 : {
 72 :     int step = 0;
 73 :     int stm_dim = 0;
 74 :
 75 :     init();                            // 内蔵周辺機能の初期化
 76 :
 77 :     motor( DCM_STOP | stm_data[0] );   // モータ初期出力
 78 :
 79 :     while( 1 ) {
 80 :         if( TCSW == 1 ) {              // タクトスイッチ ON なら
 81 :             if( TGSW == 0 ) {          // トグルスイッチ OFF なら
 82 :                 // ステッピングモータを時計回りに回す
 83 :                 step = 96;             // 1 回転 96 ステップ
 84 :                 while( step-- ) {
 85 :                     // 時計回りは励磁データを引いていく
 86 :                     stm_dim--;
 87 :                     if( stm_dim < 0 ) stm_dim = 7;
 88 :                     // ステッピングモータに次の励磁データをセット
 89 :                     motor( DCM_STOP | stm_data[stm_dim] );
 90 :                     timer_ms( 50 );    // 進むまで待つ
 91 :                 }
 92 :             } else {                   // トグルスイッチ ON なら
 93 :                 // ステッピングモータを反時計回りに回す
 94 :                 step = 96;
 95 :                 while( step-- ) {
 96 :                     // 反時計回りは励磁データを足していく
 97 :                     stm_dim++;
 98 :                     if( stm_dim > 7 ) stm_dim = 0;
 99 :                     // ステッピングモータに次の励磁データをセット
100 :                     motor( DCM_STOP | stm_data[stm_dim] );
101 :                     timer_ms( 50 );    // 進むまで待つ
102 :                 }
103 :             }
104 :         }
105 :     }
106 : }
107 :
108 : ////////////////////////////////////////////////////////////////////////
109 : // end of file
110 : ////////////////////////////////////////////////////////////////////////
```

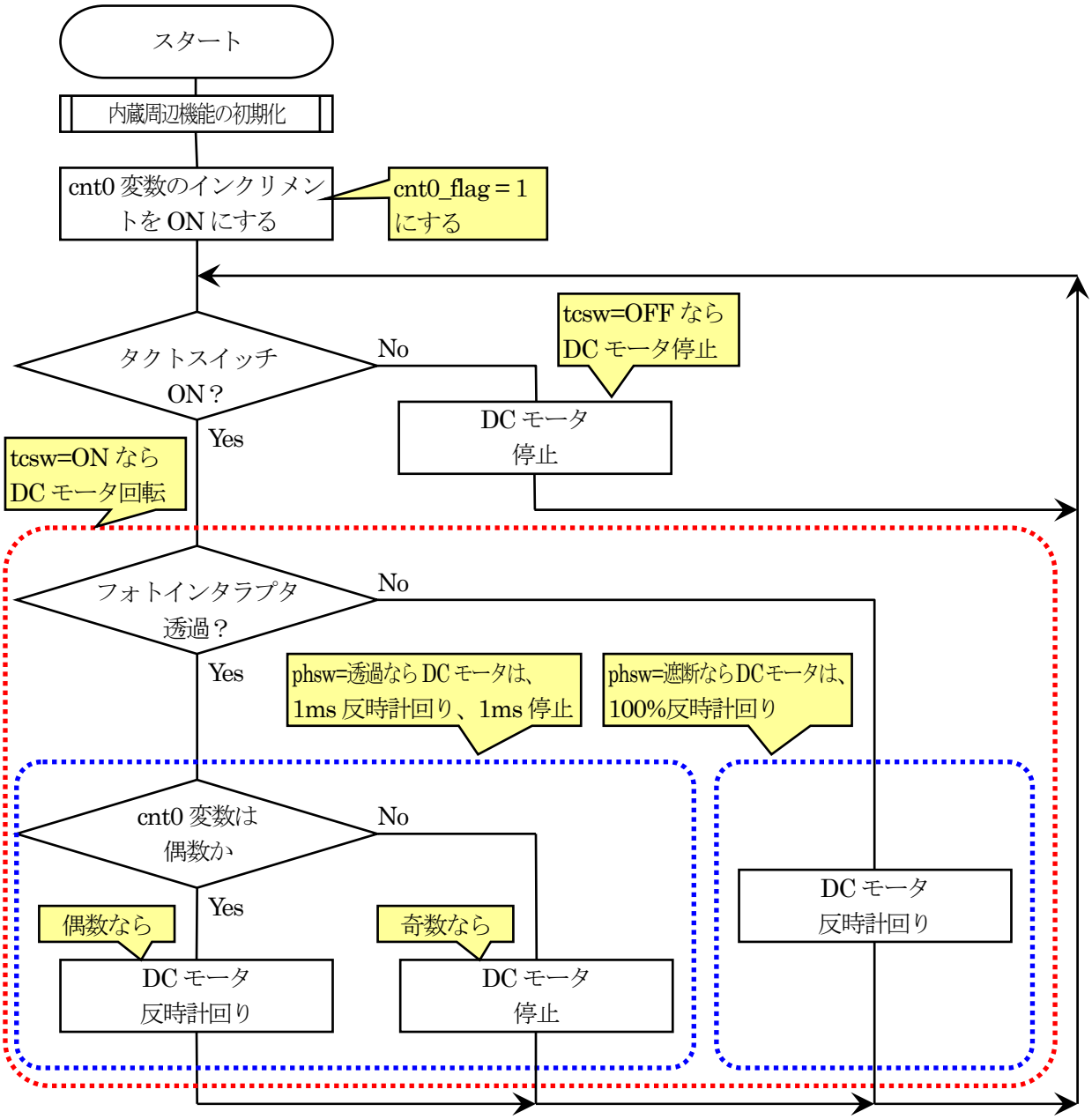
4.11　課題 3

4.11.1　課題

(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。
　(ア)　フォトインタラプタが「透過」で、低速に回転する。
　(イ)　フォトインタラプタが「遮断」で、高速に回転する。
　(ウ)　上記(ア)と(イ)は回転中にいつでも変更ができる。

▲大会当日配付資料より抜粋

4.11.2　フローチャート



4.11.3 プログラム例

```
 1 : ////////////////////////////////////////////////////////////////////////
 2 : // 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　課題３
 3 : // 座席番号：●
 4 : // 氏　　名：○○　○○
 5 : //
 6 : // Copyright (C) 2011 ルネサスマイコンカーラリー事務局
 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 :     cnt0_flag = 1;                    // cnt0 変数のインクリメントを ON にする
22 :
23 :     while( 1 ) {
24 :         if( tcsw == 1 ) {             // タクトスイッチ ON なら
25 :             if( phsw == 0 ) {         // フォトインタラプタ 透過なら
26 :                 // DC モータ　50%で反時計回り
27 :                 if( cnt0 % 2 == 0 ) {   // cnt0 変数の値で回転、停止させる
28 :                     dcm_dir = M_HANTOKEI;   // DC ﾓｰﾀ：反時計回り
29 :                 } else {
30 :                     dcm_dir = M_STOP;       // DC ﾓｰﾀ：停止
31 :                 }
32 :             } else {
33 :                 // DC モータ 100%で反時計回り
34 :                 dcm_dir = M_HANTOKEI;       // DC ﾓｰﾀ：反時計回り
35 :             }
36 :         } else {
37 :             // タクトスイッチ OFF なら、DC モータ停止
38 :             dcm_dir = M_STOP;               // DC ﾓｰﾀ：停止
39 :         }
40 :     }
41 : }
42 :
43 : ////////////////////////////////////////////////////////////////////////
44 : // end of file
45 : ////////////////////////////////////////////////////////////////////////
```

4.11.4 プログラムの解説

行	詳細
21	cnt0 変数は 1ms ごとに自動で+1 する変数です。cnt0_flag を 0 以外の値にすることで、割り込みプログラムで＋1 するようになります。
26～31	フォトインタラプタが透過なら、DC モータを低速で回転させます。cnt0 変数が偶数なら反時計回りに、奇数なら停止させます。具体的には cnt0 変数を 2 で割って余りがなければ偶数と判断します。モータは 1ms 反時計回り、1ms 停止を繰り返すので、100%回転させたときの約 1/2 の速さで回転します。
34	フォトインタラプタが遮断なら、DC モータを反時計回りさせます。
38	タクトスイッチが OFF なら、DC モータを停止させます。

4.11.5 割り込み、共通ファイルを使わないプログラム例「kadai3_kanni.c」

割り込み、共通ファイル(common.c)を使わないプログラム例を下記に示します。動作は、使用したときのフローチャートと同様です。

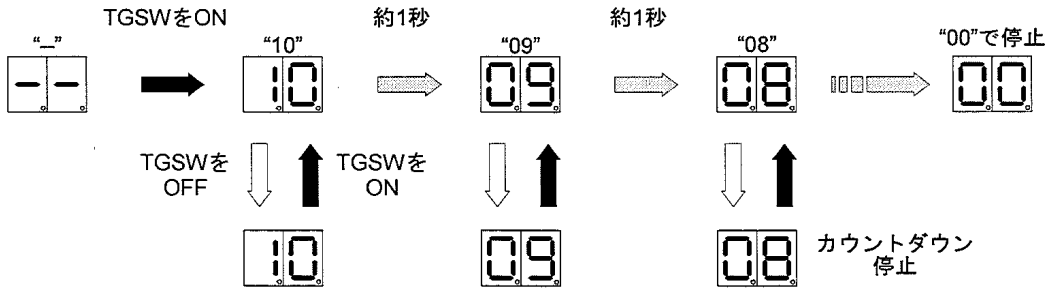
```
 1 :  ////////////////////////////////////////////////////////////////////////
 2 :  // 第 11 回高校生ものづくりコンテスト全国大会　電子回路組立部門　課題３
 3 :  //                                          （共通ファイル、割り込み　未使用版）
 4 :  // 座席番号：●
 5 :  // 氏　　名：○○　○○
 6 :  //
 7 :  // Copyright (C) 2011 ルネサスマイコンカーラリー事務局
 8 :  ////////////////////////////////////////////////////////////////////////
 9 :
10 :  ////////////////////////////////////////////////////////////////////////
11 :  // インクルード
12 :  ////////////////////////////////////////////////////////////////////////
13 :  #include "sfr_r838a.h"                  // R8C/38A SFR の定義ファイル
14 :  #include "r8c38a_lib.h"                 // R8C/38A マイコンライブラリ
15 :
16 :  ////////////////////////////////////////////////////////////////////////
17 :  // シンボル定義
18 :  ////////////////////////////////////////////////////////////////////////
19 :  #define PHSW            p0_0            // フォトインタラプタ
20 :  #define TCSW            p0_1            // タクトスイッチ
21 :
22 :  #define STM_STOP        0x0f            // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ:ｽﾄｯﾌﾟ
23 :  #define DCM_STOP        0x00            // DC ﾓｰﾀ:ｽﾄｯﾌﾟ
24 :  #define DCM_HANTOKEI    0x10            // DC ﾓｰﾀ:反時計回り
25 :
26 :  ////////////////////////////////////////////////////////////////////////
27 :  // R8C/38A スペシャルファンクションレジスタ(SFR)の初期化
28 :  ////////////////////////////////////////////////////////////////////////
29 :  void init( void )
30 :  {
31 :      // CPU の動作クロックを XIN クロックにする
32 :      init_xin_clk();
33 :
34 :      // ポートの入出力設定
35 :      pd( 0 ,0xf8 );                      // 7-6:回路③ 2-0:回路①
36 :      pd( 1, 0xf0 );                      // 5:RXD0 4:TXD0 3-0:DIP SW
37 :      pd( 2 ,0xff );                      // 7-0:回路③
38 :      pd( 3 ,0xff );
39 :      pd( 4 ,0xb8 );                      // 7:XOUT 6:XIN 5:LED 2:VREF
40 :      pd( 5 ,0xff );
41 :      pd( 6 ,0xff );
42 :      pd( 7 ,0xff );
43 :      pd( 8 ,0xff );
44 :      pd( 9 ,0xff );
45 :  }
46 :
47 :  ////////////////////////////////////////////////////////////////////////
48 :  // ステッピングモータと DC モータの制御
49 :  ////////////////////////////////////////////////////////////////////////
50 :  void motor( unsigned char data )
51 :  {
52 :      p0_6 = 1;                           // 左側 7 セグ 消灯
53 :      p0_7 = 1;                           // 右側 7 セグ 消灯
54 :
55 :      p2   = data;                        // 74HC564 の入力端子に出力
56 :      p2_7 = 1;                           // 74HC564 の CK 立ち上げ→モータ出力
57 :      p2_7 = 0;
58 :  }
59 :
```

```
60 : ////////////////////////////////////////////////////////////////////////
61 : // メイン関数
62 : ////////////////////////////////////////////////////////////////////////
63 : void main( void )
64 : {
65 :     init();                            // 内蔵周辺機能の初期化
66 :
67 :     while( 1 ) {
68 :         if( TCSW == 1 ) {              // タクトスイッチ ON なら
69 :             if( PHSW == 0 ) {          // フォトインタラプタ 透過なら
70 :                 // DC モータ  50%で反時計回り
71 :                 // 反時計回りと停止を 1ms ごとに繰り返して低速にする
72 :                 motor( STM_STOP | DCM_HANTOKEI );
73 :                 timer_ms( 1 );
74 :                 motor( STM_STOP | DCM_STOP );
75 :                 timer_ms( 1 );
76 :             } else {                   // フォトインタラプタ 透過なら
77 :                 // DC モータ 100%で反時計回り
78 :                 motor( STM_STOP | DCM_HANTOKEI );
79 :             }
80 :         } else {                       // タクトスイッチ OFF なら
81 :             // タクトスイッチ OFF なら、DC モータ停止
82 :             motor( STM_STOP | DCM_STOP );
83 :         }
84 :     }
85 : }
86 :
87 : ////////////////////////////////////////////////////////////////////////
88 : // end of file
89 : ////////////////////////////////////////////////////////////////////////
```

4.12 課題 4

4.12.1 課題

(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」で、10 進数で"10"から"00"までのカウントダウンを行う。
　(ア) カウントダウンの状態を左右の 7 セグメント LED に表示する。
　(イ) 約 1 秒ごとにカウントダウンし、"00"で停止する。
(3) カウントダウン中にトグルスイッチが「OFF」で、カウントダウンを停止する。
　(ア) 停止中の 7 セグメント LED の表示は、停止した直後の表示を維持する。
　(イ) 停止中に再びトグルスイッチが「ON」で、そこからカウントダウンを引き続き行う。



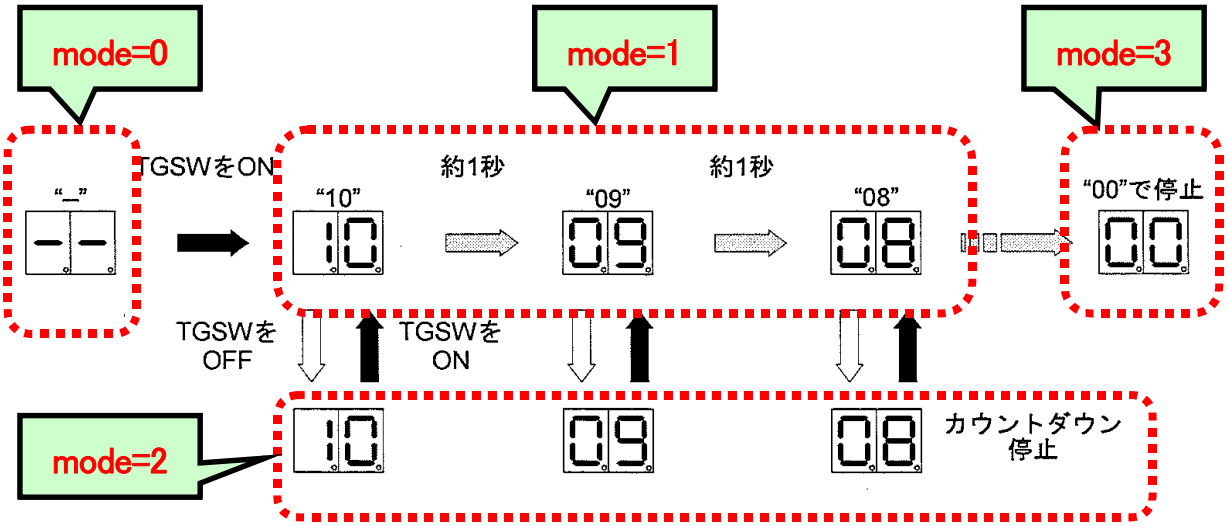
▲大会当日配付資料より抜粋

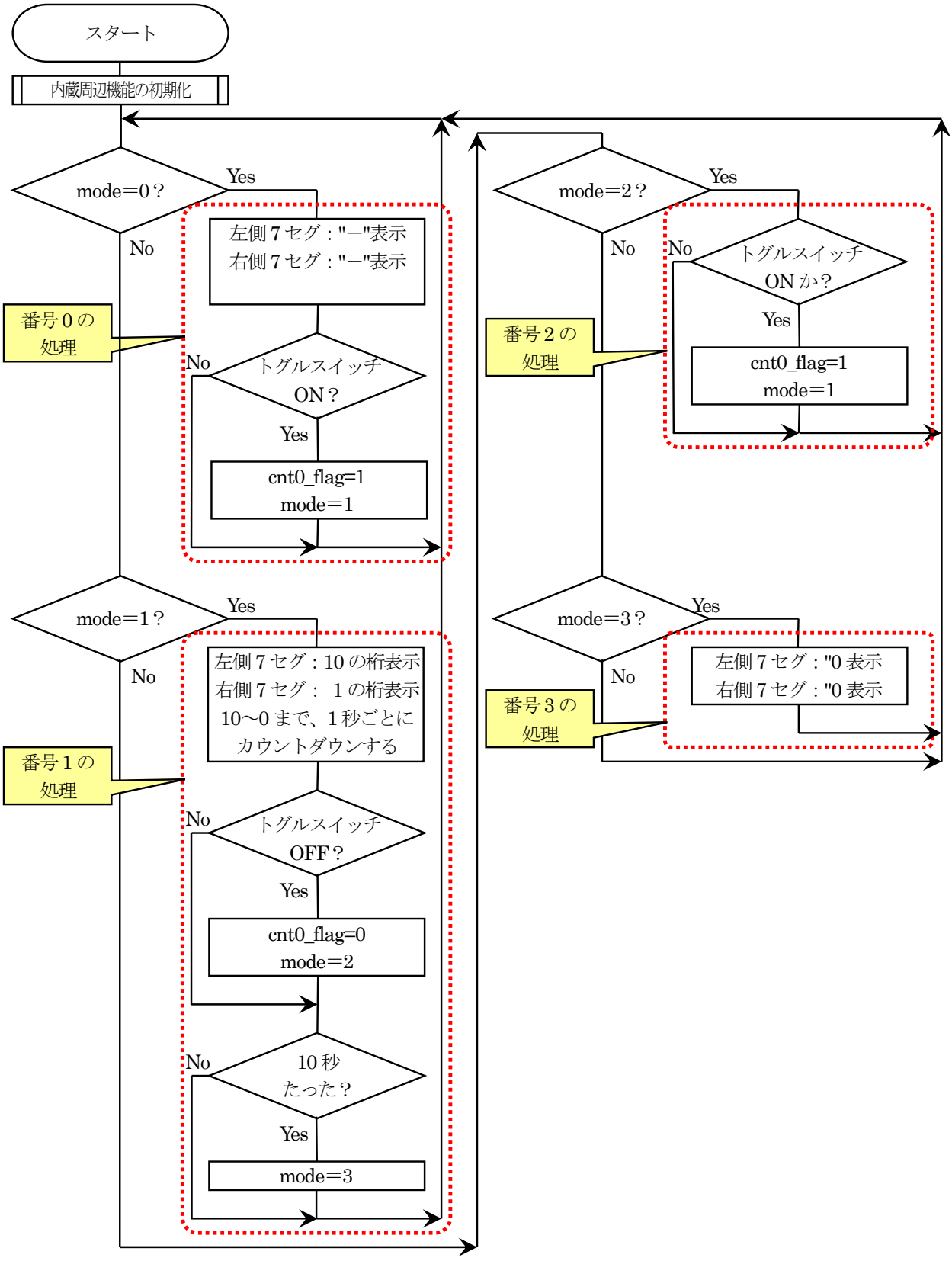
4.12.2　フローチャート

今回は、それぞれの処理に番号を付けて、番号ごとに区切ってプログラムを作ります。実際のプログラムでは、mode 変数を宣言して、この変数の値を処理番号とし、switch-case 文で処理を分けます。

処理番号とプログラムを、下表に示します。

番号	状態	左 7 セグ	右 7 セグ	処理番号が変わる条件
0	下図の「mode=0」の部分	″_″表示	″_″表示	・タクトスイッチ ON なら 1 番へ移る
1	下図の「mode=1」の部分	″10″～″01″まで 1 秒ごとに カウントダウン		・タクトスイッチ OFF なら 2 番へ移る ・10 秒たったら 3 番に移る
2	下図の「mode=2」の部分	前の値を維持	前の値を維持	・タクトスイッチが ON なら 1 番に移る
3	下図の「mode=3」の部分	″0″表示	″0″表示	




スタート
内蔵周辺機能の初期化
mode=0？
Yes
No
左側 7 セグ："−"表示
右側 7 セグ："−"表示
番号 0 の処理
No
トグルスイッチ ON？
Yes
cnt0_flag=1
mode=1
mode=1？
Yes
No
左側 7 セグ：10 の桁表示
右側 7 セグ：　1 の桁表示
10〜0 まで、1 秒ごとに
カウントダウンする
番号 1 の処理
No
トグルスイッチ OFF？
Yes
cnt0_flag=0
mode=2
No
10 秒 たった？
Yes
mode=3
mode=2？
Yes
No
No
トグルスイッチ ON か？
Yes
番号 2 の処理
cnt0_flag=1
mode=1
mode=3？
Yes
No
左側 7 セグ："0 表示
右側 7 セグ："0 表示
番号 3 の処理

4.12.3 プログラム例

```
 1 : ////////////////////////////////////////////////////////////////////////
 2 : // 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　課題４
 3 : // 座席番号：●
 4 : // 氏　　名：○○　○○
 5 : //
 6 : // Copyright (C) 2011 ルネサスマイコンカーラリー事務局
 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 :     int mode = 0;                     // 動作モード
20 :
21 :     init();                           // 内蔵周辺機能の初期化
22 :
23 :     while( 1 ) {
24 :         switch( mode ) {
25 :             case 0:
26 :                 // プログラム開始直後
27 :                 seg_left  = SEG_MAI;    // 開始直後の７セグ表示
28 :                 seg_right = SEG_MAI;
29 :                 if( tgsw == 1 ) {       // トグルスイッチ ON なら
30 :                     cnt0_flag = 1;
31 :                     mode = 1;
32 :                 }
33 :                 break;
34 :
35 :             case 1:
36 :                 // カウントダウン
37 :                 seg_left  = (10- cnt0 / 1000) / 10; // 10 の桁
38 :                 seg_right = (10- cnt0 / 1000) % 10; //  1 の桁
39 :                 if( tgsw == 0 ) {       // トグルスイッチ OFF なら
40 :                     cnt0_flag = 0;
41 :                     mode = 2;
42 :                 }
43 :                 if( cnt0 >= 10000 ) {   // 10 秒たったなら
44 :                     cnt0_flag = 0;
45 :                     mode = 3;
46 :                 }
47 :                 break;
48 :
49 :             case 2:
50 :                 // トグルスイッチ ON になるまで待機
51 :                 if( tgsw == 1 ) {       // トグルスイッチが ON なら
52 :                     cnt0_flag = 1;
53 :                     mode = 1;
54 :                 }
55 :                 break;
56 :
57 :             case 3:
58 :                 // 10 秒たったなら"00"表示で終了
59 :                 seg_left  = 0;
60 :                 seg_right = 0;
61 :                 break;
62 :         }
63 :     }
64 : }
65 :
66 : ////////////////////////////////////////////////////////////////////////
67 : // end of file
68 : ////////////////////////////////////////////////////////////////////////
```

4.12.4 プログラムの解説

行	詳細
24	mode 変数の値の確認は、switch-case 文を使いました。if 文でも構いません。
26～33	番号 0 の処理です。 開始直後は、左側 7 セグメント LED、右側 7 セグメント LED ともに″－″を表示します。タクトスイッチが ON になったら番号を 1 にします。
36～47	番号 1 の処理です。 左側 7 セグメント LED、右側 7 セグメント LED に″10″～″01″まで、1 秒ごとにカウントダウンしていきます。実際は、cnt0 変数を使って表示します。この変数は 1ms ごとに+1 するので次の処理を行っています。 ① cnt0 変数を 1000 で割って、1 秒ごとに＋1するようにする ② ①の値を 10 で引いて、10 から 1 秒ごとに**カウントダウン**するようにする ③ ②の値を 10 で割って（10 の桁を求めて）、左側 7 セグメント LED に表示する ④ ②の値を 10 で割った余りを求めて、右側 7 セグメント LED に表示する トグルスイッチが OFF になったら、cnt0_flag を 0 にして、番号を 2 にします。 cnt0 変数が 10000 以上、すなわち 10 秒たったなら cnt0_flag を 0 にして、番号を 3 にします。
50～55	番号 2 の処理です。 トグルスイッチが ON になるまでこの部分で待機します。 トグルスイッチが ON になったなら cnt0_flag 変数を 1 にして、番号を 1 にします。
58～61	番号 3 の処理です。 左側 7 セグメント LED、右側 7 セグメント LED に″00″を表示して何もしません。ここで終了です。

4.13 課題 5

4.13.1 課題

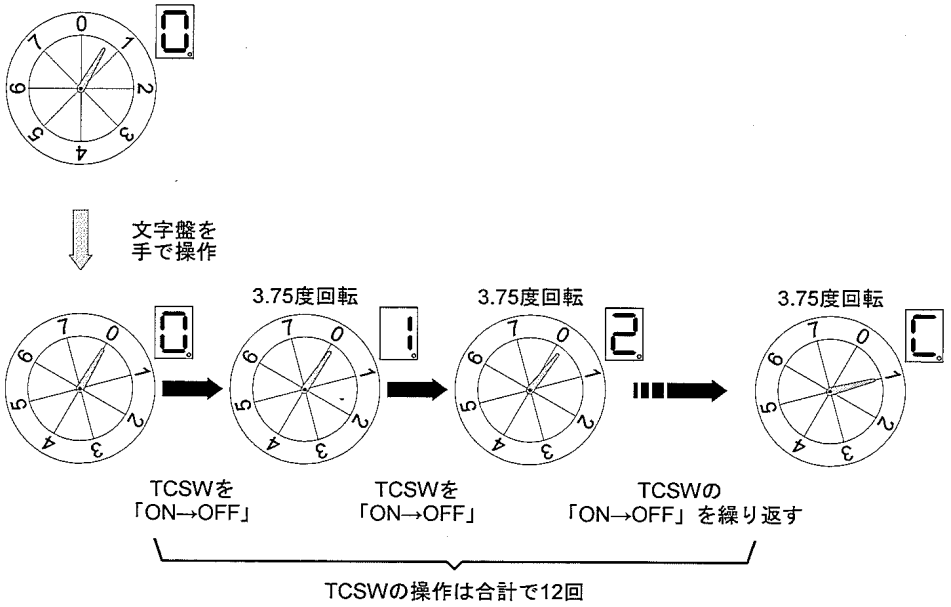
(1) プログラム開始直後、右側の 7 セグメント LED に"0"を表示する。その後、ステッピングモータの目盛盤を手で動かし 0 の位置を指示針に合わせる。

(2) タクトスイッチの「ON→OFF」を 12 回行い、ステッピングモータを時計回りに 45 度回転させる。

(ア) 1 回のタクトスイッチの「ON→OFF」で、ステッピングモータは 3.75 度回転する。

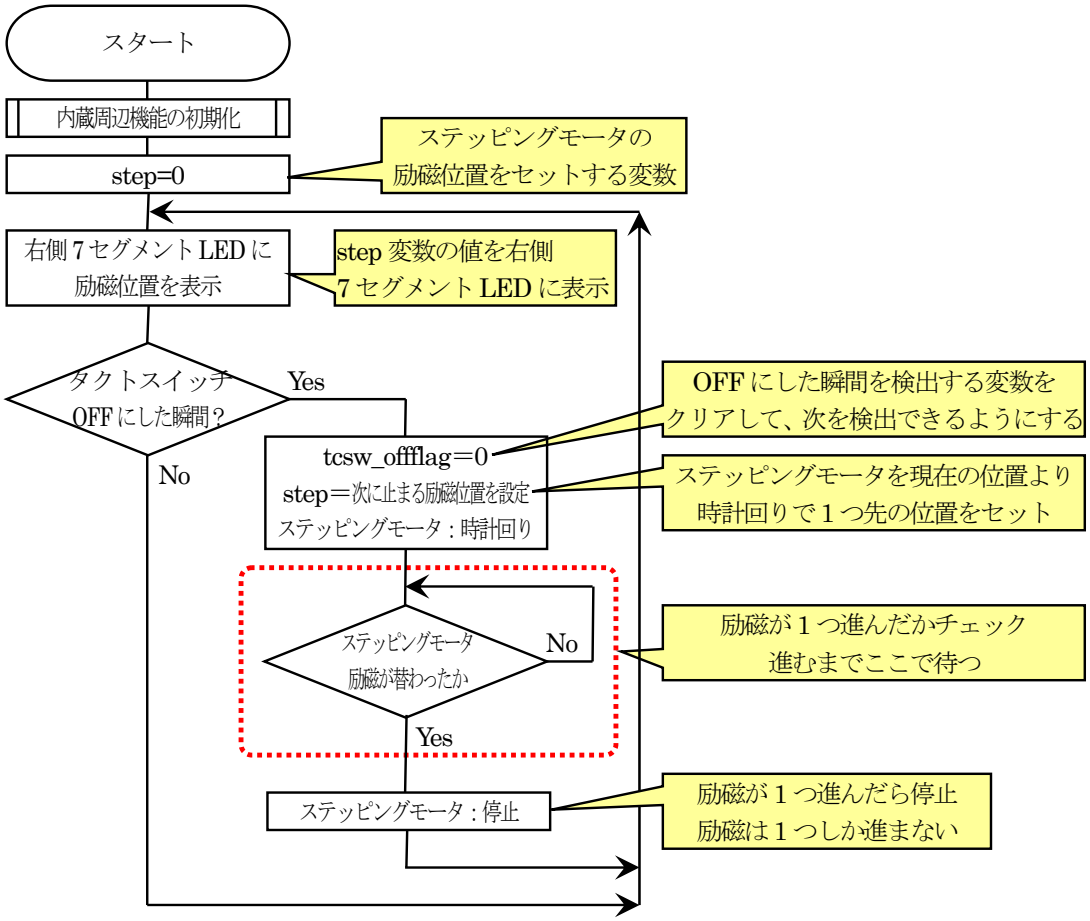
(イ) 右側の 7 セグメント LED に、タクトスイッチの操作回数を 16 進数で表示する。

(ウ) 上記(ア)と(イ)は、タクトスイッチの「ON→OFF」の「OFF」の瞬間に動作する。



▲大会当日配付資料より抜粋

4.13.2　フローチャート



4.13.3 プログラム例

```
 1 : ////////////////////////////////////////////////////////////////////////
 2 : // 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　課題５
 3 : // 座席番号：●
 4 : // 氏　　名：○○　○○
 5 : //
 6 : // Copyright (C) 2011 ルネサスマイコンカーラリー事務局
 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 :     int step = 0;                     // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀの励磁位置
20 :
21 :     init();                           // 内蔵周辺機能の初期化
22 :
23 :     while( 1 ) {
24 :         seg_right = step;             // 右側 7 セグに励磁位置を表示
25 :
26 :         if( tcsw_offflag == 1 ) {     // タクトスイッチ ON→OFF なら
27 :             tcsw_offflag = 0;
28 :
29 :             step = stm_step_cnt + 1;  // 次の励磁位置を記憶
30 :             stm_dir = M_TOKEI;        // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ：時計回り
31 :
32 :             while( stm_step_cnt != step );  // 励磁が替わるまで待つ
33 :             stm_dir = M_STOP;         // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ：停止
34 :         }
35 :     }
36 : }
37 :
38 : ////////////////////////////////////////////////////////////////////////
39 : // end of file
40 : ////////////////////////////////////////////////////////////////////////
```

4.13.4 プログラムの解説

行	詳細
19	step 変数は、ステッピングモータの励磁位置をセットする変数です。右側 7 セグメント LED の表示用にも使います。
24	右側 7 セグメント LED に励磁位置を表示します。
26	タクトスイッチが ON から OFF になった瞬間を検出します。 tcsw_offflag 変数が 1 になったら、ON から OFF になった瞬間です。
29～30	step 変数に、現在の励磁位置より1つ進んだ励磁位置をセットし、ステッピングモータを時計回りに回します。
32～33	step 変数の位置までステッピングモータが励磁したかチェックします。励磁するまで 32 行を繰り返します。励磁したならステッピングモータを停止させます。

4.14 課題 6

4.14.1 課題

(1) タクトスイッチが「ON」の瞬間に回転しているモータがあれば、モータを停止し、「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。

（ア）「ON」にしてから「OFF」になるまでの時間が約 1 秒未満なら、ステッピングモータも DC モータも回転しない。

（イ）「ON」にしてから「OFF」になるまでの時間が約 1 秒以上 3 秒未満なら、ステッピングモータのみが反時計回りに回転する。

（ウ）「ON」にしてから「OFF」になるまでの時間が約 3 秒以上なら、DC モータのみが反時計回りに回転する。

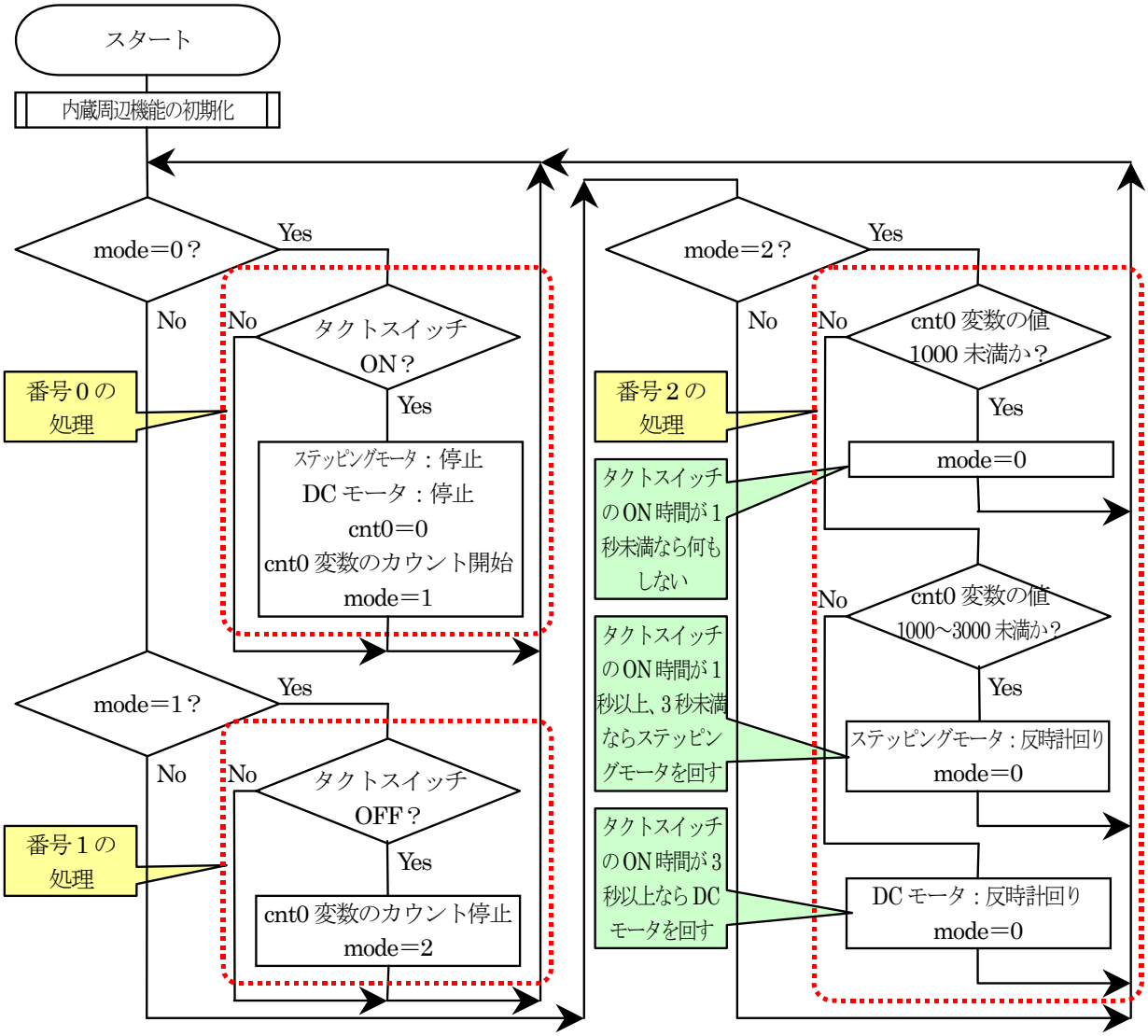
▲大会当日配付資料より抜粋

4.14.2 フローチャート

今回は、それぞれの処理に番号を付けて、番号ごとに区切ってプログラムを作ります。実際のプログラムでは、mode 変数を宣言して、この変数の値を処理番号とし、switch-case 文で処理を分けます。

処理番号とプログラムを、下表に示します。

番号	状態	ステッピングモータ	DC モータ	処理番号が変わる条件
0	「mode=0」の部分	停止	停止	・タクトスイッチが ON なら 1 番へ移る
1	「mode=1」の部分	停止	停止	・タクトスイッチが OFF なら 2 番へ移る
2	「mode=2」の部分	タクトスイッチの ON 時間が 1～3 秒なら反時計回り	タクトスイッチの ON 時間が 3 秒以上なら反時計回り	・タクトスイッチの ON 時間に応じた動作をしたら 0 番へ移る


スタート
内蔵周辺機能の初期化
mode＝0？
Yes
No
No
タクトスイッチ
ON？
Yes
番号 0 の
処理
ステッピングモータ：停止
DC モータ：停止
cnt0＝0
cnt0 変数のカウント開始
mode＝1
mode＝1？
Yes
No
No
タクトスイッチ
OFF？
Yes
番号 1 の
処理
cnt0 変数のカウント停止
mode＝2
mode＝2？
Yes
No
No
cnt0 変数の値
1000 未満か？
Yes
番号 2 の
処理
mode＝0
タクトスイッチ
の ON 時間が 1
秒未満なら何も
しない
No
cnt0 変数の値
1000～3000 未満か？
Yes
タクトスイッチ
の ON 時間が 1
秒以上、3 秒未満
ならステッピン
グモータを回す
ステッピングモータ：反時計回り
mode＝0
タクトスイッチ
の ON 時間が 3
秒以上なら DC
モータを回す
DC モータ：反時計回り
mode＝0

4.14.3 プログラム例

```
 1 : ////////////////////////////////////////////////////////////////////////
 2 : // 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　課題６
 3 : // 座席番号：●
 4 : // 氏　　名：○○　○○
 5 : //
 6 : // Copyright (C) 2011 ルネサスマイコンカーラリー事務局
 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 :     int mode = 0;                      // 動作モード
20 :
21 :     init();                            // 内蔵周辺機能の初期化
22 :
23 :     while( 1 ) {
24 :         switch( mode ) {
25 :             case 0:
26 :                 // タクトスイッチ ON を検出
27 :                 if( tcsw == 1 ) {      // タクトスイッチ ON なら
28 :                     stm_dir = M_STOP;  // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ：停止
29 :                     dcm_dir = M_STOP;  // DC モータ：停止
30 :                     cnt0 = 0;          // タクトスイッチ ON 時間計測変数クリア
31 :                     cnt0_flag = 1;     // cnt0 のカウント開始
32 :                     mode = 1;
33 :                 }
34 :                 break;
35 :
36 :             case 1:
37 :                 // タクトスイッチ OFF を検出
38 :                 if( tcsw == 0 ) {
39 :                     cnt0_flag = 0;
40 :                     mode = 2;
41 :                 }
42 :                 break;
43 :
44 :             case 2:
45 :                 // タクトスイッチの ON 時間によって制御
46 :                 if( cnt0 < 1000 ) {
47 :                     // 1 秒未満なら STM、DCM、回転しない
48 :                     mode = 0;
49 :                 } else if( cnt0 >= 1000 && cnt0 < 3000 ) {
50 :                     // 1 秒秒以上、3 秒未満なら STM のみ反時計回りに回転
51 :                     stm_dir = M_HANTOKEI;   // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ：反時計回り
52 :                     mode = 0;
53 :                 } else {
54 :                     // 3 秒秒以上なら DCM のみ反時計回りに回転
55 :                     dcm_dir = M_HANTOKEI;   // DC ﾓｰﾀ：反時計回り
56 :                     mode = 0;
57 :                 }
58 :                 break;
59 :         }
60 :     }
61 : }
62 :
63 : ////////////////////////////////////////////////////////////////////////
64 : // end of file
65 : ////////////////////////////////////////////////////////////////////////
```

4.14.4 プログラムの解説

行	詳細
26～34	番号 0 の処理です。 タクトスイッチが押されたら、 ①ステッピングモータを停止 ②DC モータを停止 ③cnt0 変数(1ms ごとに+1 する変数)の値をクリア ④cnt0 のカウント開始 をします。その後、番号を 1 にします。 ①②は、課題の『**「(1)タクトスイッチが「ON」の瞬間に回転しているモータがあれば、モータを停止し、**』のために入れています。1 回目はモータが停止しているので入れる必要は無いですが、2 回目以降は必要です。
37～42	番号 1 の処理です。 タクトスイッチが OFF になるまで待ちます。 OFF になったら、cnt0 変数のカウントを停止して番号を 2 にします。cnt0 変数には、タクトスイッチを ON していた時間(押していた時間)が代入されます。
45～58	番号 2 の処理です。 タクトスイッチを押していた時間によって、処理が替わります。 ON の時間は、cnt0 変数の値で分かります。 ON の時間が 1 秒未満なら、すなわち cnt0 変数が 1000 未満なら、何もしません。 ON の時間が 1 秒以上 3 秒未満なら、すなわち cnt0 変数が 1000 以上 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。 ON の時間が 3 秒以上なら、すなわち cnt0 変数が 3000 以上なら DC モータを反時計回りに回します。 その後、番号を 0 にして、再度タクトスイッチが ON になるのを待ちます。

4.15 課題 7

4.15.1 課題

課題 7 の問題を、下記に示します。

(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目盛盤を手で動かし 0 の位置を指示針に合わせる。

(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。

(ア) 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。

(イ) 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。

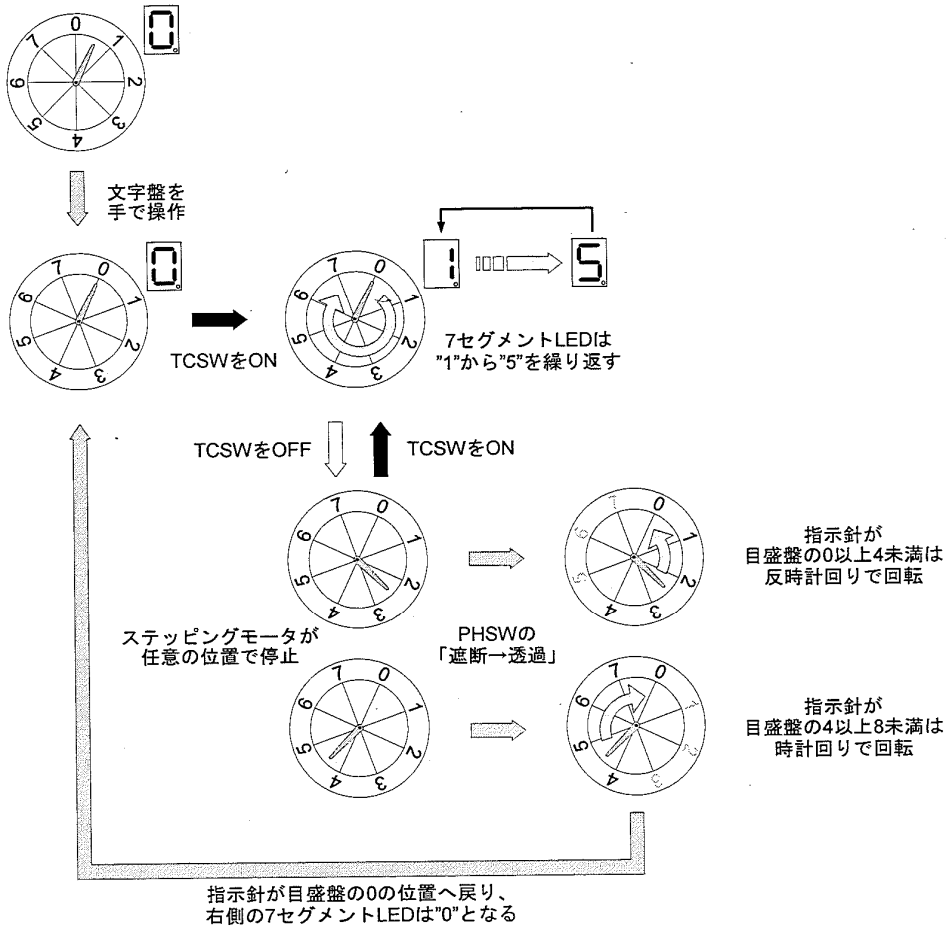
(ウ) 右側の 7 セグメント LED の表示は、"5"の次は"1"に戻り繰り返す。

(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」すると、**「遮断」の瞬間**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。

(ア) ステッピングモータの指示針が目盛盤の 0 以上 4 未満の場合は、反時計回りに回転する。

(イ) ステッピングモータの指示針が目盛盤の 4 以上 8(0 の位置)未満の場合は、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 の位置で戻ると、右側の 7 セグメント LED の表示は"0"となり、(2)以降の動作が行える。



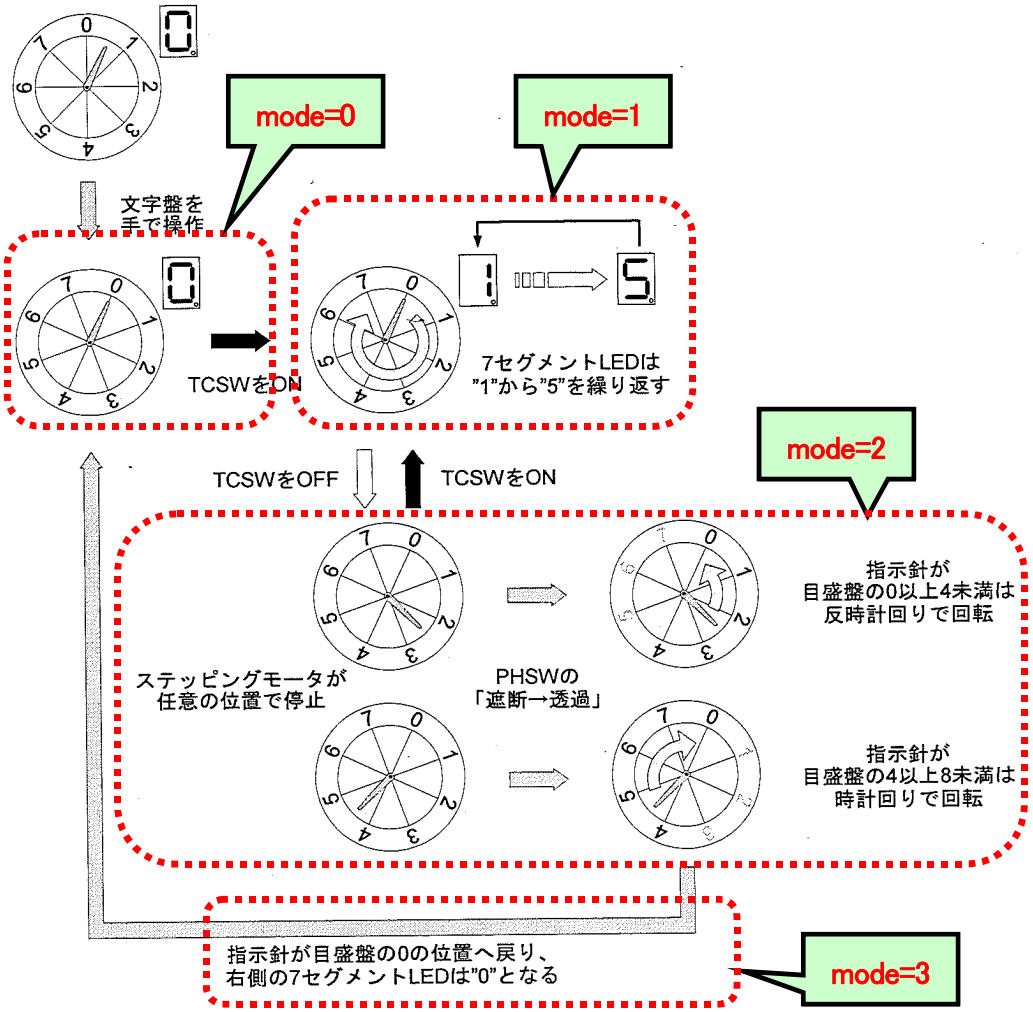
▲大会当日配付資料より抜粋

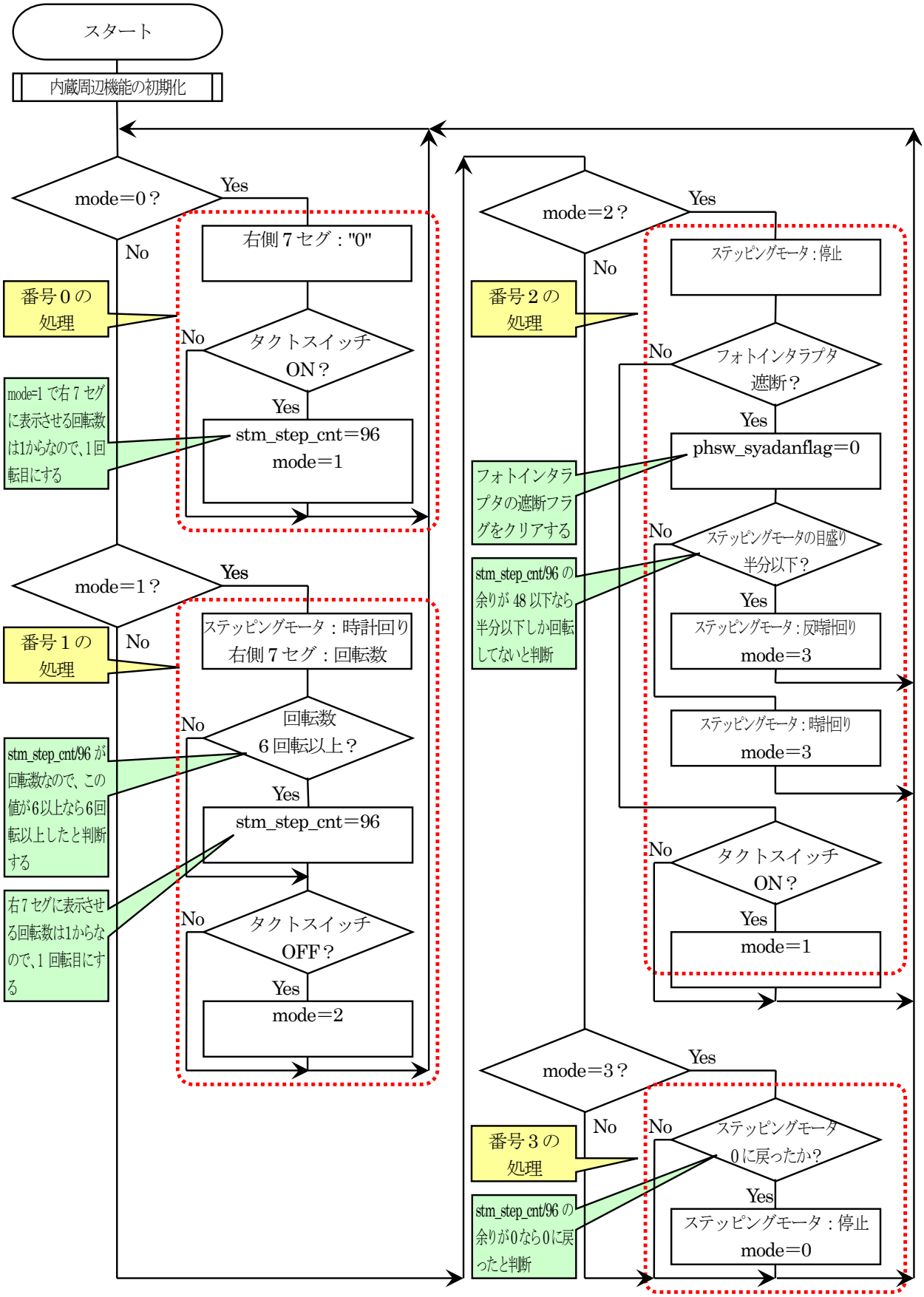
4.15.2　フローチャート

今回は、それぞれの処理に番号を付けて、番号ごとに区切ってプログラムを作ります。実際のプログラムでは、mode 変数を宣言して、この変数の値を処理番号とし、switch-case 文で処理を分けます。

処理番号とプログラムを、下表に示します。

番号	状態	ステッピング モータ	右側 7 セグメント LED	処理番号が変わる条件
0	図の「mode=0」の部分	停止	″0″表示	・タクトスイッチが ON になったら 1 番へ移る
1	図の「mode=1」の部分	時計回り	ステッピングモータの回転数に応じて″1″～″5″を表示 ″5″の次は″1″に戻る	・タクトスイッチが OFF ならに 2 番へ移る
2	図の「mode=2」の部分	停止	前の表示を維持	・フォトインタラプタが透過から遮断になった瞬間に 3 番へ移る
3	図の「mode=3」の部分	・目盛盤が 0 以上 4 未満なら反時計回り ・目盛盤が 4 以上 8 未満なら時計回り	前の表示を維持	・ステッピングモータの指示針が 0 になったら停止させて 0 番へ移る




スタート
内蔵周辺機能の初期化
mode=0？
Yes
No
右側 7 セグ："0"
番号 0 の処理
タクトスイッチ ON？
No
Yes
mode=1 で右 7 セグに表示させる回転数は1からなので、1 回転目にする
stm_step_cnt=96
mode=1
mode=1？
Yes
No
番号 1 の処理
ステッピングモータ：時計回り
右側 7 セグ：回転数
回転数 6 回転以上？
No
Yes
stm_step_cnt/96 が回転数なので、この値が6以上なら6回転以上したと判断する
stm_step_cnt=96
右 7 セグに表示させる回転数は1からなので、1 回転目にする
タクトスイッチ OFF？
No
Yes
mode=2
mode=2？
Yes
No
番号 2 の処理
ステッピングモータ：停止
フォトインタラプタ遮断？
No
Yes
phsw_syadanflag=0
フォトインタラプタの遮断フラグをクリアする
ステッピングモータの目盛り半分以下？
No
Yes
stm_step_cnt/96 の余りが 48 以下なら半分以下しか回転してないと判断
ステッピングモータ：反時計回り
mode=3
ステッピングモータ：時計回り
mode=3
タクトスイッチ ON？
No
Yes
mode=1
mode=3？
Yes
No
番号 3 の処理
ステッピングモータ 0 に戻ったか？
No
Yes
stm_step_cnt/96 の余りが0なら0に戻ったと判断
ステッピングモータ：停止
mode=0

4.15.3 プログラム例

```
 1 : ////////////////////////////////////////////////////////////////////////
 2 : // 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　課題 7
 3 : // 座席番号：●
 4 : // 氏　　名：○○　○○
 5 : //
 6 : // Copyright (C) 2011 ルネサスマイコンカーラリー事務局
 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 :     int mode = 0;                      // 動作モード
20 :
21 :     init();                            // 内蔵周辺機能の初期化
22 :
23 :     while( 1 ) {
24 :         switch( mode ) {
25 :             case 0:
26 :                 // プログラム開始直後
27 :                 seg_right = 0;
28 :                 if( tcsw == 1 ) {      // タクトスイッチ ON なら
29 :                     stm_step_cnt = 96; // 表示は"1"からなので 1 周したことにする
30 :                     mode = 1;
31 :                 }
32 :                 break;
33 :
34 :             case 1:
35 :                 // ステッピングモータを回す
36 :                 stm_dir   = M_TOKEI;    // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ：時計回り
37 :                 seg_right = stm_step_cnt / 96;  // 右側 7 セグに回転数を表示
38 :                 if( stm_step_cnt / 96 >= 6 ) {
39 :                     // 周回数が 6 回転以上なら 1 にする
40 :                     stm_step_cnt = 96;
41 :                 }
42 :                 if( tcsw == 0 ) {      // タクトスイッチ OFF なら
43 :                     mode = 2;
44 :                 }
45 :                 break;
46 :
47 :             case 2:
48 :                 // タクトスイッチ OFF のときステッピングモータ停止
49 :                 stm_dir = M_STOP;              // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ：停止
50 :                 if( phsw_syadanflag == 1 ) {   // フォトインタラプタ遮断なら
51 :                     phsw_syadanflag = 0;
52 :                     if( stm_step_cnt % 96 < 48 ) {
53 :                         // ステッピングモータの目盛りが半分以下(0～47)なら
54 :                         stm_dir = M_HANTOKEI;  // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ：反時計回り
55 :                         mode = 3;
56 :                         break;
57 :                     } else {
58 :                         // ステッピングモータの目盛りが半分以上(48～95)なら
59 :                         stm_dir = M_TOKEI;     // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ：時計回り
60 :                         mode = 3;
61 :                         break;
62 :                     }
63 :                 }
64 :                 if( tcsw == 1 ) {              // タクトスイッチ ON なら
65 :                     mode = 1;
66 :                 }
67 :                 break;
68 :
```

```
69 :                case 3:
70 :                    // ステッピングモータが 0 になったか
71 :                    if( stm_step_cnt % 96 == 0 ) {
72 :                        stm_dir = M_STOP;           // ｽﾃｯﾋﾟﾝｸﾞﾓｰﾀ：停止
73 :                        mode = 0;
74 :                    }
75 :                    break;
76 :            }
77 :        }
78 :  }
79 :
80 :  ////////////////////////////////////////////////////////////////////////////
81 :  // end of file
82 :  ////////////////////////////////////////////////////////////////////////////
```

4.15.4 プログラムの解説

行	詳細
26～32	番号 0 の処理です。 タクトスイッチが ON になったなら stm_step_cnt を 96 にします。これは mode=1 で右側 7 セグメント LED に表示させる回転数は1からなので、1 回転させたことにするためです。その後、処理番号を 1 番にします。
35～45	番号 1 の処理です。 右側 7 セグメント LED には回転数を表示します。stm_step_cnt／96 が回転数なので、この値を右側 7 セグメント LED に出力します。 右側 7 セグメント LED に表示する回転数は、5 の次は 1 にします。よって stm_step_cnt／96 が 6 以上なら stm_step_cnt=96 として、1 回転目ということにします。 タクトスイッチが OFF なら処理番号を 2 番にします。
48～67	番号 2 の処理です。 ステッピングモータを停止します。 フォトインタラプタが遮断された瞬間を検出したら、次の動作を行い処理番号を 3 番にします。 ・ステッピングモータの目盛りが半分未満ならステッピングモータを反時計回りにします。半分以下かどうかは、stm_step_cnt／96 の余りが 48 未満なら回転は半分未満と判断します。 ・ステッピングモータの目盛りが半分以上ならステッピングモータを時計回りにします。半分以上かどうかは、stm_step_cnt／96 の余りが 48 以上なら回転は半分以上と判断します。 タクトスイッチが ON になったなら、処理番号を 1 にします。
70～75	番号 3 の処理です。 処理番号 2 で、ステッピングモータは反時計回りか時計回りしています。ここではステッピングモータの目盛りが 0 になったことを検出して、回転を停止させます。 0 になったかどうかは、stm_step_cnt/96 の余りが 0 なら、目盛りが 0 にきたと判断できます。余りが 0 になったならステッピングモータの回転を停止させて処理番号を 0 番します。

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
プログラム解説マニュアル

発行年月日　　2012 年 2 月 1 日　　第 1.00 版

発行　　(株)ルネサスソリューションズ　ルネサスマイコンカーラリー事務局
〒162-0824　東京都新宿区揚場町 2-1　軽子坂MNビル