平成19年度 (3年生前期)

実験実習

Experiments in Electrical Engineering

― H8マイコン実験I ―

秋田工業高等専門学校　　電気情報工学科

作成日　2007年5月20日
作成者　山本昌志

目 次

諸注意

以下，注意事項を箇条書きするので，厳守すること．これは，山本が担当している「H8 マイコン実験 I」にのみ適用する．

1. 実験時の服装など
   - PC ルームで実験を行うので，清潔な服装であること．作業服の必要はない．
   - 履物は，PC ルーム備え付けのサンダルを使うこと．
2. 実験ノート
   - A4 の実験ノートを用意すること．ルーズリーフは不可である．
   - 実験ノートへの記述は，ボールペンあるいは万年筆とする．鉛筆は不可である．
3. レポートの提出
   - 2 日目の実験日から，毎回提出すること．実験時寒中にチェックを行い，不備な点の再提出を求める．
   - 次回の実験日の AM8:45 までとする．それ以降に提出したものは，減点の対象とする．
   - 前期あるいは後期の最後の実験の場合は，その 1 週間後とする．
   - 提出期限に遅れたものは，減点とする．
   - 未提出のレポートがある場合は，単位を与えることができないので注意すること．
   - 提出先は，担当教官のレポート入れとする．
4. レポートの再提出
   - 内容に不備があるものはレポートの再提出を課す．
   - 再提出の期限は，再提出を言い渡された 1 週間以内とする．
5. レポートの書き方
   - ガイダンスのときには配布した「実験実習のレポートの書き方」を参考にすること．
6. その他
   - 実験に関する資料は，web(www.ipc.akita-nct.ac.jp/~yamamoto/) に載せてある．必要に応じて参考にするのがよいであろう．
   - 実験を休んだ場合，放課後を利用して，実験を行うこと．実験を行う場合，機材を貸し出すので，申し出ること．
   - 放課後，実験を行うこともできる．その場合は，担当教員 (山本) に申し出ること．

実験テーマ 303

H8 マイコン実験 I

1 目的

ルネサステクノロジ社のマイコン H8 を使って，機器の制御実験の基本を学習する．LED やモーター，スピーカーなどを C 言語を使ったマイコンの制御を学ぶ．さらに，A/D 変換，液晶モニターへの表示方法も学習する．

2 原理

2.1 マイコン

メモリーや周辺回路、CPU を内蔵した産業用の LSI を総称して，ワンチップ・マイコン— 略してマイコン[1]— と呼ぶ．ひとつのパッケージの中に，コンピューターの動作に必要な回路が全て組み込まれている．これひとつで，独立したコンピューターとなっており，プログラムを書くことにより，様々な動作が可能である．そのため，機器の制御に向いており，家電製品を中心に広く使われている．

マイコンは，CPU とメモリー，I/O ポート—入出力ポート—から成っている．その様子を図 1 にしめす．I/O ポートを一つのチップに内蔵することにより，容易に機器の制御が可能となっている．制御の内容はプログラムにより記述が可能なため，複雑な処理ができる．

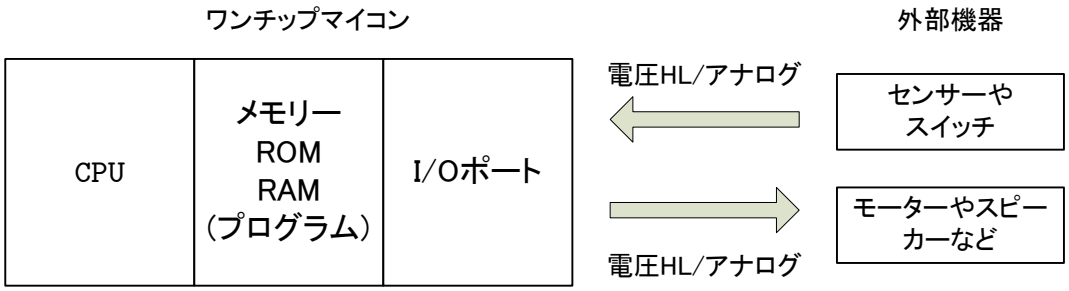

図 1: マイコンの概念図

3 実験方法

3.1 日程

この実験は，4 回 (150 分× 4) で行い，スケジュールは以下のとおりである．

1 日目
- ブレッドボードの使い方
- ポート出力実験．主に，マイコンを使って，LED を点灯させる．
- 割り込み制御実験.

2 日目
- PWM 実験.

[1] もともとは，マイクロコンピュータ (microcomputer) の略

3 日目
- AD/DA 変換器の実験
- 液晶表示装置を使った表示実験

4 日目
- 応用回路実験．自分で回路とプログラムを考えて，動作させる．

3.2 準備

3.2.1 機材

ここでは，班でひとつの実験を行うのではなく，ひとりひとりが個別に H8 の動作を調べる．表 1 に，ひとり当たり必要な実験器材を示す．表を見て分かるとおり，パソコン (PC) を使うため，電気棟 3F の PC ルームで実験を行う．

表 1: 実験器材

装置	メーカー	型番/仕様	数量
パソコン			1
H8 マイコンボード	秋月電子通商	AKI-H8/3664N	1
ブレッドボード			1
直流電源		5V 出力	1
		9V 出力	1
赤色 LED			6
トランジスター	東芝	2SC1815	2
抵抗		3.3 [kΩ]	8
セラミックコンデンサー		0.1 [μF]	2
スピーカー			1
モーター			1
液晶表示モジュール			1
温度センサー			1
OP アンプ			1
タクトスイッチ			2
ジャンパーピン			2
D-SUB コネクター			1
ブレッドボード配線材			

3.2.2 注意

実験をはじめる前に以下の注意を読み，正しく機器を使うこと．

- 指定以上の電圧を加えないこと．高い電圧を加えると，半導体は簡単に壊れる．
- LED やトランジスター，H8 マイコンの出力端子には電流制限抵抗を付けて，グランドと接続している．直接電圧を印加すると最大定格以上の電流がながれ，半導体が破壊される．回路図のとおり，抵抗を接続すること．

- LED やダイオード，トランジスターには極性があり，正しく接続する必要がある．図 2 や図 4 に，それぞれの極性を表す．
- H8 マイコンへのプログラム転送の手順を間違えないこと．
- プログラムが暴走したら，[Ctrl] に続けて [c] を押すことにより，強制終了させる．
- ここでの実験は，順を追って回路を組み上げる．回路のレイアウトをきちんとしないと，最後の方で配線が大変になる．図 5 で示すようにブレッドボードに回路を配置すれば，きれいに仕上る．

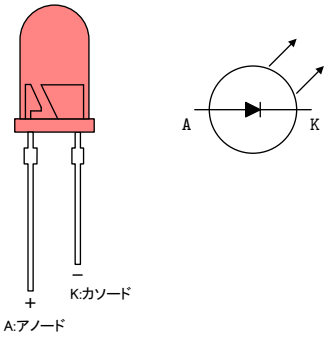

図 2: LED 外観と記号

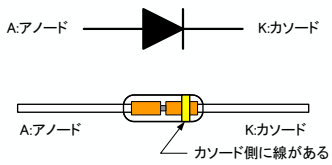

図 3: ダイオードの外観と記号

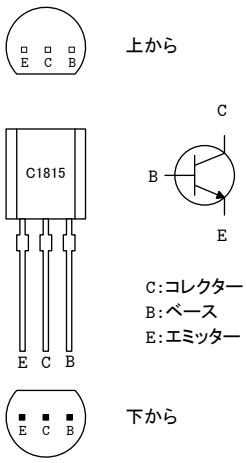

図 4: トランジスターの外観と記号

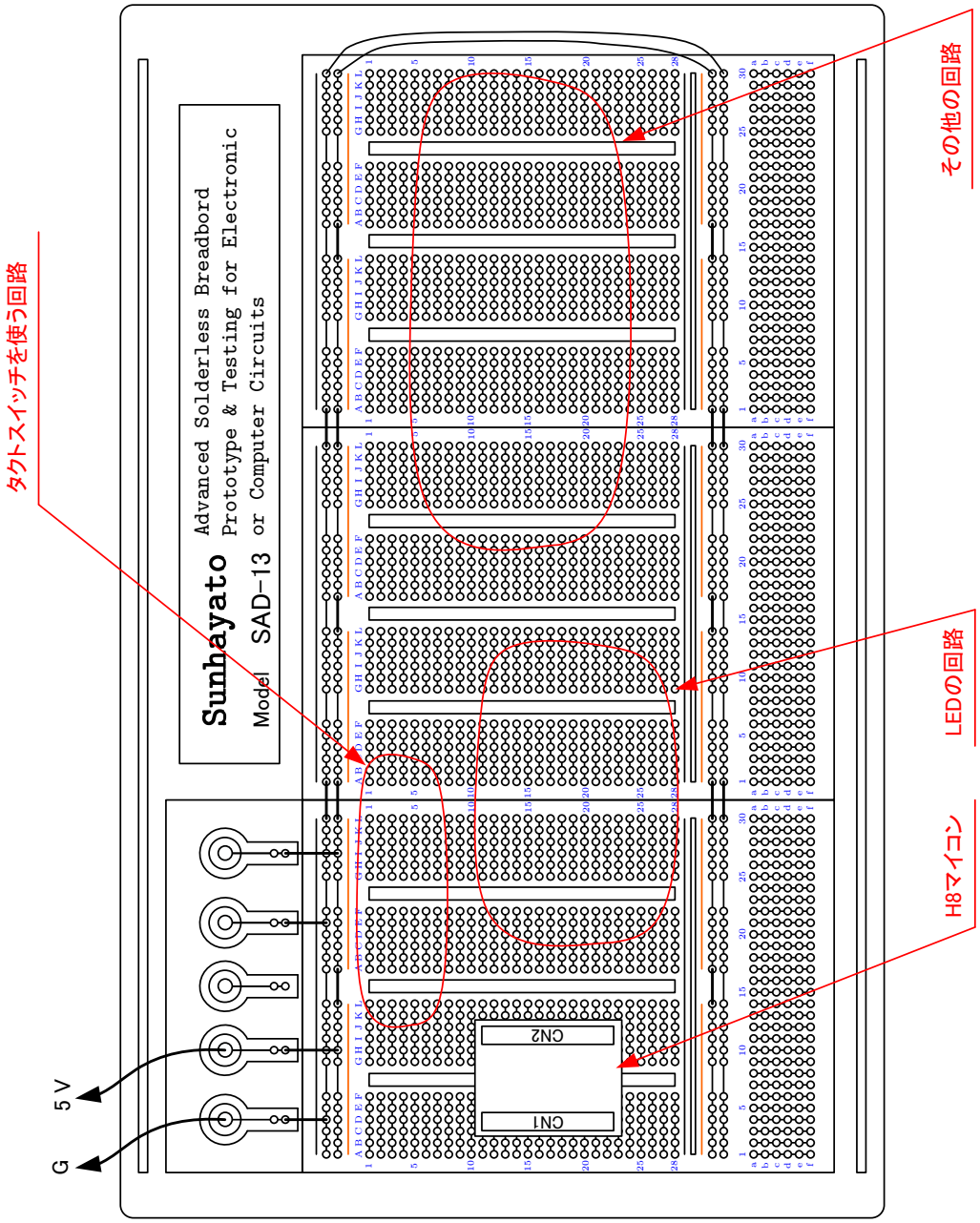

図 5: ブレッドボード上の配置

3.2.3 ブレッドボードについて

ここでは，回路をブレッドボード (bread bord)[2]上に作成する (図 5)．通常，回路の作成にはハンダ付の作業が伴う．しかし，ブレッドボードを使うとその作業が不要となり，回路の変更が容易である．学生実験や回路のテストを行う場合，とても都合が良い．

ブレッドボードを見て分かるように，たくさんの小さな穴が開いている．穴の間隔は 1/10 インチとなっており，それは IC(Integrated Circuit) の足の間隔に等しい．この穴に IC を差し込んで，回路を作成する．

[2]ブレッドボードとは，もともとパンをこねる板のことである．昔々—たぶん真空管の時代—電子回路をテストするために，この板の上でバラックの回路を組んだことがはじまりと推測できる．

IC に限らず，抵抗やスイッチ，トランジスター等の半導体部品も差し込むことができる．差し込んだ部品は配線材により接続して，回路とする．

ブレッドボードを使うためには，内部の配線を理解しなくてはならない．ここで使うブレッドボードは，主に 4 つのブロックからできあがっている．そのひとつの内部配線を図 6 にしめす．数個の穴が内部で，電気的に接続されている．この接続を理解して，ブレッドボードを上手に使おう．

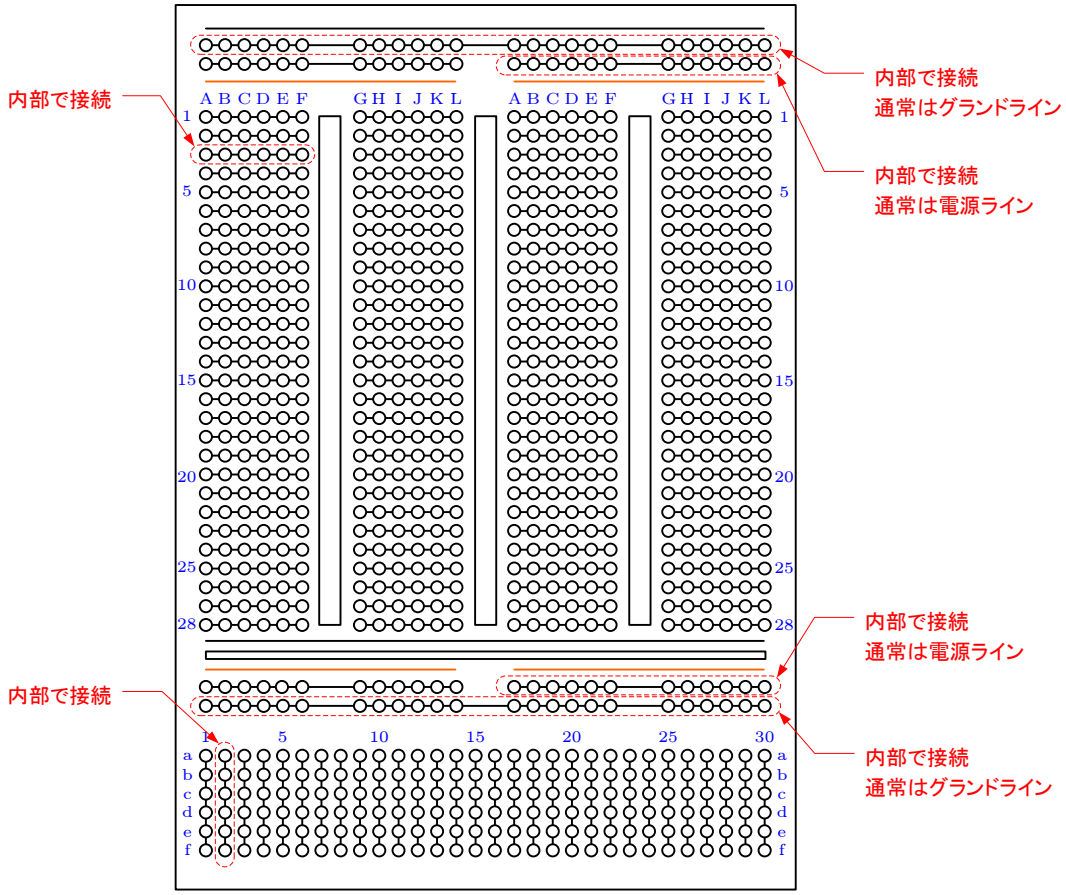

図 6: ブレッドボードの内部配線

[**練習 1**] テスターを使ってブレッドボード内部での配線を確認せよ．

3.2.4 ソフトウェアーについて

この実験のプログラムは，gcc を使って開発する．gcc は統合開発環境ではないため，プログラム開発は原始的な部分[3]から作成しなくてはならない．原始的な部分からプログラムを開発すると大変ではあるが，コンピューターに関する多くの知識が得られる．この実験では原始的な部分に触れないが，もしこれに興味があれば参考文献 [1] から読みはじめればよいだろう．また，私の WEB ページ「H8 プログラム開発 (Linux C 言語 編)[4]」でも記述している．

この実験で必要な原始的な部分のプログラムは全て，ダウンロードして使う．このプログラムの中身を全て理解できれば，一人前である．興味のある者はこれを理解するよう努力せよ．コンピューターの仕組みがよくわかるようになる．

[3]アセンブラやリンカースクリプト，スタートアップルーチンを指す．
[4]http://www.akita-nct.jp/ yamamoto/comp/H8/C_linux/C_linux.php

諸君がこの実験で記述するプログラムには原始的な部分はあまり見えないようにしているが，それでもビット操作等が必要になる．その時は，じっくり考えて動作の内容を理解しなくてはならない．

3.3 ポート出力実験

3.3.1 概要

ポートとレジスター H8 の I/O ポートの一つである Port5 を使って，LED の点灯実験を行う．プログラムにより，Port5 の電圧を H(5V) と L(0V) と変化させて，I/O ポートに接続した LED の様子を調べる．Port5 の電圧は，H8 のレジスター PDR5 で設定できる．レジスター PDR5 と I/O ポートの Port5 の電圧の関係を図 7 にしめす．

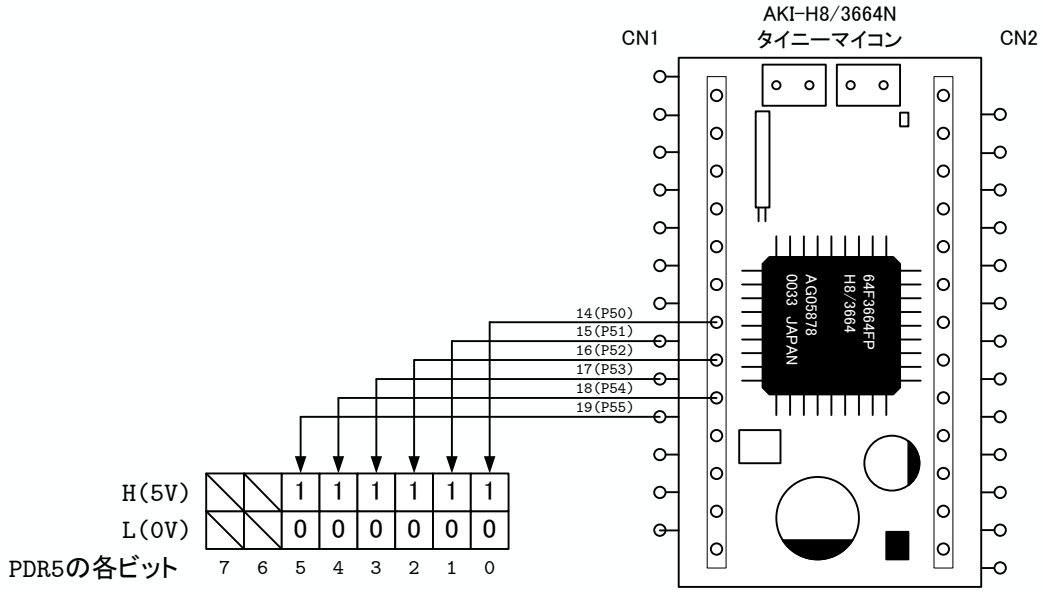

図 7: PDR5 と Port5 の電圧の関係

プログラム 図 7 から，Port5 は CN1 の 14–19 番ピンに割り当てられていることが分かる．ここに LED を接続すると，それらを点灯/消灯することができる．このピンの出力電圧はレジスター PDR5 の下位 6 ビットにより決まる (図 7)．すなわち，8 ビットのレジスターである PDR5 の第 0 ビットが 1 になると，14 番ピンに 5 [V] が出力され LED は点滅する．反対にビットが 0 になると LED は消灯する．プログラムにより，H8 マイコンの PDR5 レジスターの設定すると，そこに接続された LED を点灯/消灯できるのである．ソフトウェアーでハードウェアーを制御している．

レジスター PDR5 の値を C 言語で設定するには，次のようにする．

```
PDR5 = 0x01;
```

これで，PDR5 の第 0 ビットを 1 にして，他の第 1–7 ビットを 0 に設定している．この方法だとまずい場合がある．第 6 ビットと第 7 ビットは他の用途に使われており，それらのいっしょに変更している．第 6 ビットと第 7 ビットを変更しないで，他のビットを変更するには，次のようにする．

```
PDR5 = (0x01 & 0x3f) | (PDR5 & 0xc0);
```

このようにすると，最初の 0x01 の部分を変えることで，第 0—5 ビットを任意に設定できる．ここでは．右辺の 0x01 により，PDR5 レジスターの第 0 ビットに 1 を設定している．もし，ここを 0xff としても，第 0—5 ビットは 1 が設定されるが，第 6 と 7 ビットは変化しない．このようなことができる理由を以下に示す．

- 0x3f は，下位 6 ビットのみ変更可能としている．0x3f はマスクと呼ばれるもので，そのビットパターンは 00111111 となる．&は論理積 (AND) である．0x01 & 0x3f は，下位 6 ビットを設定しているのである．
- PDR5 & 0xc0 で，上位 2 ビットのビットパターンはそのままで，下位 6 ビットをゼロにしている．PDR5 & 0xc0 は，上位 2 ビットを設定しているのである．
- 最終的に，これらのビットの論理和 (OR) をとることにより，8 ビットの設定ができる．|が論理和の演算である．

練習問題　プログラムを理解したならば，次の練習問題を実施せよ．

［**練習 2**］ 式を使って，PDR5 = (0x01 & 0x3f) | (PDR5 & 0xc0) とする理由を説明せよ．

3.3.2　1 つの LED 点灯実験

はじめに，ひとつの LED の点灯と消灯の実験を行う．プログラムを変えることにより，LED の制御を行う．

プログラム　リスト 1 のプログラムの 8 行目で，レジスター PDR5 の第 0 ビットを 1 に設定している．そのため，このプログラムを実行すると CN1 の 14 番ピンから 5 [V] が出力される．すると，そこに接続された LED が点灯する．0x01 を 0x00 にすると，接続した LED は点灯しない．

実験順序　PDR5 の第 0 ビットの値と CN1 の 14 番ピンの出力の関係を以下の実験で調べる．

1. 図 9 のような回路をブレッドボード上に作成する．回路を作成しているとき，その部品の破損を防ぐために電源を接続していはならない．意図しない電流が素子に流れることがあるからである．図 5 の配置に従い，H8 マイコンや LED，スイッチ類を配線すること．
   (a) ブレッドボード上に電源ラインを作成する．ここでの実験使う機器は全て 5 [V] で動作する．
   (b) ブレッドボード上に H8 を取り付ける．ブレッドボードに取り付けた場合の H8 のピン番号は，図 8 のとおりである．
   (c) 残りの配線を行う．LED には正負があるので間違えないこと．
2. 配線に間違いの無いことをチェックする．
3. プログラムに必要なファイルをダウンロードする．ファイル名や拡張子を変えてはならない．Makefile をダウンロードするときに余分な拡張子—.htm や.html— が付くことがあるので，取り除くこと．
4. リスト 1 のとおり H8 マイコンのプログラムを作成する．
   - ファイル名は，「experiment.c」とする．
   - コメント文は，記述しなくてもよい．

- ダウンロードしたファイルは，変更してはならない．

5. コンパイルする．コマンドは,「make」である．

6. make の結果，できあがったプログラムを H8 マイコンへ転送する．

   (a) H8 マイコン基板の JP2 と JP3 をジャンパーピンでショートする．

   (b) 回路に 5 [V] を供給する; 電源のバナナジャックの正 (赤) をブレッドボードに接続する).

   (c) コマンド「make write」をタイプし，プログラムを転送する．転送がはじまると，「H8/3664F is ready! 2001/2/1 Yukio Mituiwa.」と表示される．もし失敗したならば，[Ctrl] と [c] を押して，プログラムを止める．

   (d) 転送には，20 秒くらい必要である．「EEPROM Writing is successed.」と表示されるまで待つ．

7. H8 マイコンを実行させる．

   (a) 5 [V] の電源を OFF にする; 電源のバナナジャックの正 (赤) をブレッドボードから引き抜く．

   (b) H8 マイコン基板の JP2 と JP3 のジャンパーピンを取り外し，オープンにする．

   (c) ブレッドボードに，5 [V] を供給する．

   (d) プログラムが実行されて，LED が点灯する．

   (e) プログラムを最初から実行させたければ，RES スイッチを押す．このプログラムでは状態の変化は分からない．

リスト 1: LED 点灯実験のプログラム

```
1  #include "3664.h"
2
3  int main()
4  {
5
6    init_led();
7
8    PDR5 = (0x01 & 0x3f) | (PDR5 & 0xc0);       /* PDR5は，port5のデータレジスタ */
9
10   while(1){
11     SLEEP();
12   }
13 }
```

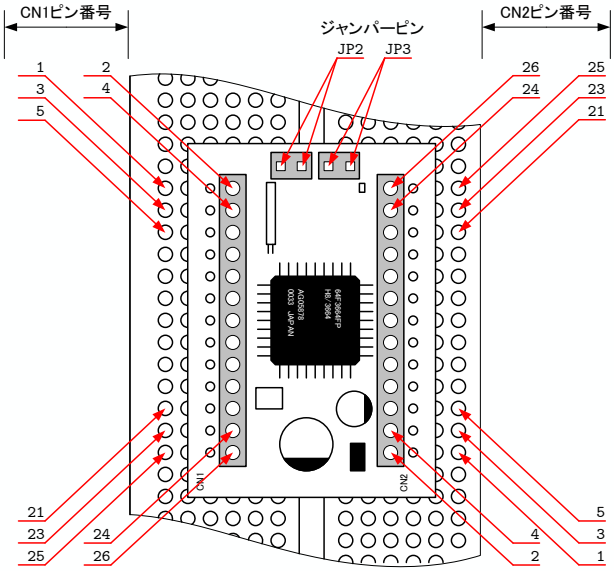

図 8: ブレッドボード上での H8 のピン番号

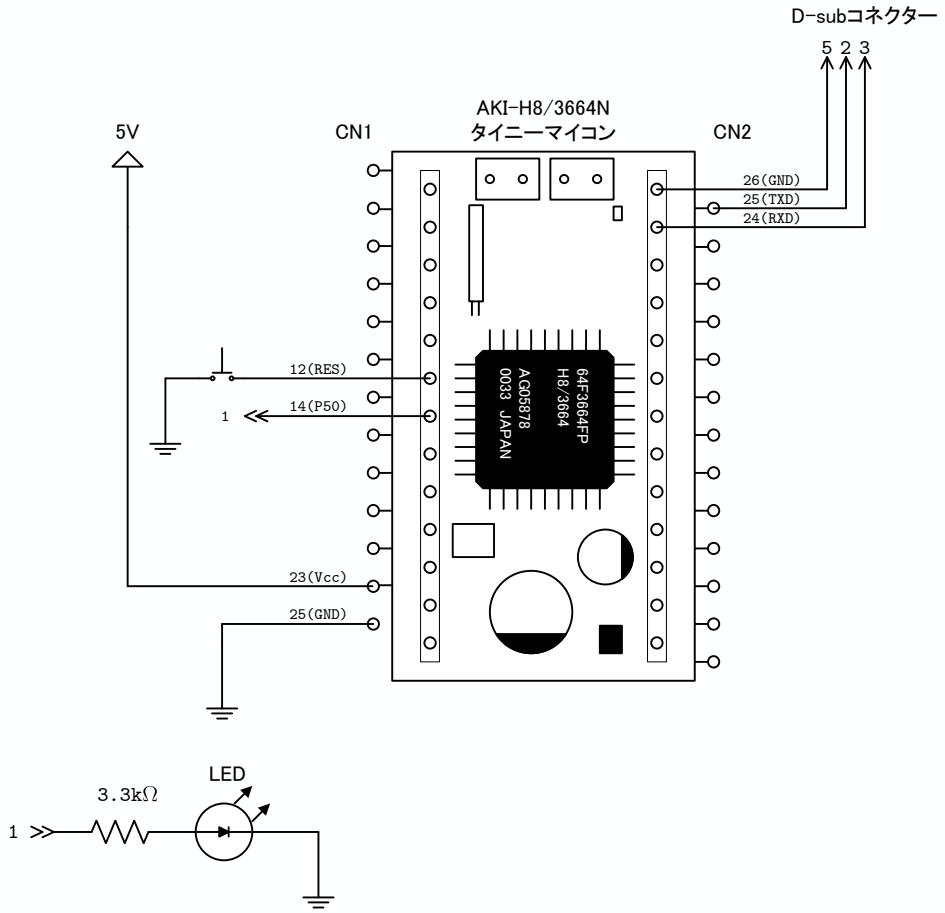

図 9: LED 点灯実験回路

[**練習 3**] テスターの電圧測定レンジを用いて，LED に流れる電流を測定せよ．電流や抵抗を測定するレンジを使ってはならない．

[**練習 4**] マイコンの出力端子に LED を直に接続してはならない．その理由を説明せよ．この実験の回路では，3.3 [KΩ] の抵抗を直列に接続している．

[**練習 5**] リスト 1 を書き換えて，LED が消灯するプログラムを作成せよ．そして，実際に動作させてみよ．

3.3.3 複数の LED 点灯実験

次に，複数の LED の点灯と消灯の実験を行う．先の実験で使ったプログラムを改良して，複数の LED の制御を行う．

プログラム リスト 2 のプログラムでは，レジスター PDR5 の下位 6 ビットの値は，0x2a から 101010 となる．このレジスターの値が CN1 の 14〜19 番ピン— P50–P55—の出力を決めている．したがって，この回路を動作させると，LED が交互に点灯する．レジスター PDR5 とポートの出力の関係は図 7 を見よ．

実験順序　先ほどの回路に LED を 5 つ追加する．さらに，プログラムの一部を書き換えて，複数個の LED の点灯/消灯の実験を行う．

1. 図 10 のような回路をブレッドボード上に作成する．ただし，電源は，まだ接続しない．
2. 配線に間違いの無いことをチェックする．
3. 先のプログラムを書き直して，リスト 2 のとおり H8 マイコンのプログラムを作成する．
   - ファイル名は，「`experiment.c`」とする．
   - コメント文は，記述しなくてもよい．
   - 他のファイルは，変更してはならない．
4. コンパイルする．コマンドは,「`make`」である．
5. `make` の結果，できあがったプログラムを H8 マイコンへ転送する．
   (a) H8 マイコン基板の JP2 と JP3 をジャンパーピンでショートする．
   (b) 回路に 5 [V] を供給する．
   (c) コマンド「`make write`」をタイプし，プログラムを転送する．転送がはじまると，「`H8/3664F is ready!  2001/2/1 Yukio Mituiwa.`」と表示される．もし失敗したならば，`[Ctrl]` と `[c]` を押して，プログラムを止める．
   (d) 転送には，20 秒くらい必要である．「`EEPROM Writing is successed.`」と表示されるまで待つ．
6. H8 マイコンを実行させる．複数の LED が転倒するはずである．
   (a) ブレッドボードから，5 [V] の電源を取り外す．
   (b) H8 マイコン基板の JP2 と JP3 のジャンパーピンを取り外し，オープンにする．
   (c) ブレッドボードに，5 [V] を供給する．
   (d) プログラムが実行されて，LED が交互に点灯する．
   (e) プログラムを最初から実行させたければ，RES スイッチを押す．このプログラムでは状態の変化は分からない．

リスト 2: LED 点灯実験のプログラム

```
1  #include "3664.h"
2
3  int main()
4  {
5
6    init_led();
7
8    PDR5 = (0x2a & 0x3f) | (PDR5 & 0xc0);       /* PDR5は，port5のデータレジスタ */
9
10   while(1){
11     SLEEP();
12   }
13 }
```

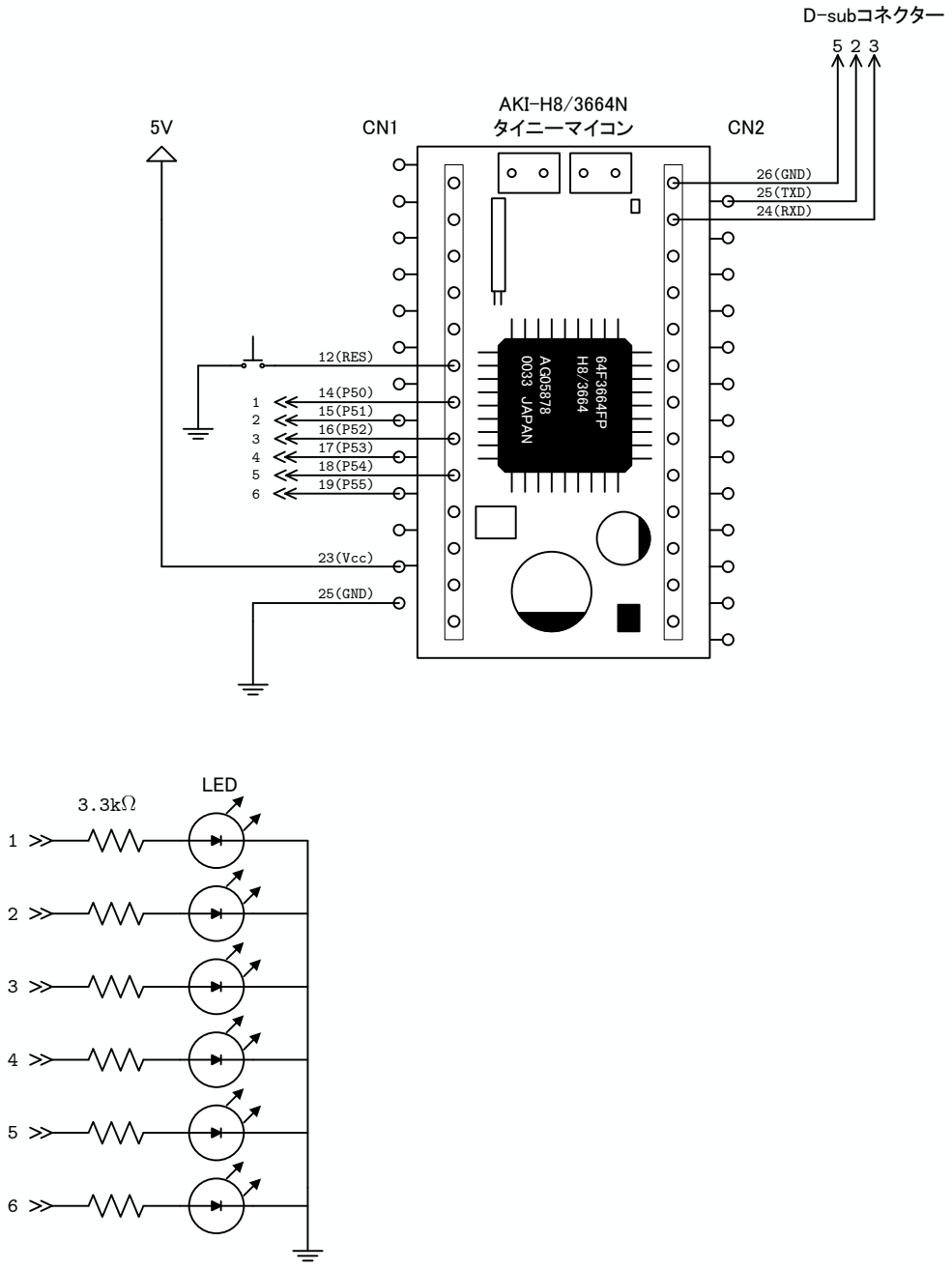

図 10: 複数 LED 点滅実験回路

[**練習 6**] リスト 2 を書き換えて，LED が ON-ON-ON-OFF-OFF-OFF となるプログラムを作成せよ．そして，実際に動作させてみよ．

[**練習 7**] さらに，リスト 2 を書き換えて，LED が ON-ON-OFF-OFF-ON-OFF となるプログラムを作成せよ．そして，実際に動作させてみよ．

3.4 割り込み制御実験

外部の信号により動作中のプログラムを止めて，他のプログラムを実行させることを割り込み処理という．この割り込み処理の実験を行う．

3.4.1 タイマー割り込み実験

実験内容 タイマー割り込みを使って，複数の LED の制御を行う．ここでのプログラムでは 0.5 秒毎に割り込みがかかり，LED の状態が変化する．接続されている 6 つの LED が 2 進数のビットパターンを表しており，それがひとつずつ増加する．

プログラム リスト 3 のプログラムでは，0.5 秒毎に割り込みがかかり，`int_timera()` 関数が実行される．すると `PDR5` の値がひとつずつ増加するので，LED の点灯状況が変わる．0.5 秒毎に 2 進数のを表す LED のパターンがひとつずつ増える．もし，`TMA` レジスターを `0x99` から `0x98` にすると 1 秒毎になる．`0x9a` にすると 0.25 秒毎，`0x9b` にすると 0.03125 秒毎になる．

実験順序

1. 実験に使う回路は，先の「複数の LED の点灯実験」と同一 (図 10) なので，変更の必要はない．
2. 先のプログラムを書き直して，H8 マイコンのプログラムを作成する．
   - 書き直すファイルは「`experiment.c`」のみである．これをリスト 3 のとおりにする．
   - コメント文は，記述しなくてもよい．
   - 他のファイルは，変更してはならない．
3. コンパイルする．コマンドは,「`make`」である．
4. `make` の結果，できあがったプログラムを H8 マイコンへ転送する．
   (a) H8 マイコン基板の JP2 と JP3 をジャンパーピンでショートする．
   (b) 回路に 5 [V] を供給する．
   (c) コマンド「`make write`」をタイプし，プログラムを転送する．転送がはじまると，「`H8/3664F is ready!  2001/2/1 Yukio Mituiwa.`」と表示される．もし失敗したならば，`[Ctrl]` と `[c]` を押して，プログラムを止める．
   (d) 転送には，20 秒くらい必要である．「`EEPROM Writing is successed.`」と表示されるまで待つ．
5. H8 マイコンを実行させる．
   - ブレッドボードから，5 [V] の電源を取り外す．
   - H8 マイコン基板の JP2 と JP3 のジャンパーピンを取り外し，オープンにする．
   - ブレッドボードに，5 [V] の電源を取り付ける．
   - プログラムを最初から実行させたければ，RES スイッチを押す．

リスト 3: タイマー割り込み実験のプログラム

```
#include "3664.h"

#pragma interrupt
void int_timera(void)
{
  CLI();
  IRR1 &= 0xbf;
  PDR5 = (PDR5++)&0x3f | (PDR5 & 0xc0);          /* カウントアップ */
  STI();
}

int main()
{

  init_led();                                    /* port5を使うときの初期化 */
  init_timer();                                  /* timer割り込みを使うときの初期化 */

  PDR5 = (0x00 & 0x3f) | (PDR5 & 0xc0);          /* PDR5は，port5のデータレジスタ */

  TMA = 0x99;

  while(1){
    SLEEP();
  }
}
```

3.4.2 IRQ 割り込み実験

実験内容 IRQ 割り込み要求 (Interrupt Request) を使って，複数の LED の制御を行う．ここでは，回路に接続したタクトスイッチを押すことにより，割り込み要求を発生させ，特定の関数を実行させる．この関数の実行により LED の状態を変える．

プログラム リスト 4 のプログラムでは，CN2 の 20 番ピン (IRQ0) に接続したタクトスイッチを押すごとに，`int_irq0()` 関数が実行される．すると `PDR5` の値がひとつずつ増加するので，LED の点灯状況が変わる．スイッチを押すごとに 2 進数のを表す LED のパターンがひとつずつ増える．

実験順序 以下実験順序を示す．注意事項等については，p.10 の「複数の LED 点灯実験」と同じである．

1. 図 11 のような回路をブレッドボード上に作成する．ただし，まだ電源は接続しない．
2. 先のプログラム「experiment.c」を書き直して，IRQ 割り込み実験プログラム「リスト 4」を作成する．
3. コンパイルする．コマンドは,「`make`」である．
4. `make` の結果，できあがったプログラムを H8 マイコンへ転送する．
5. H8 マイコンを実行させる．`IRQ0` に接続したスイッチを押すと LED の状態が変化する．

リスト 4: IRQ0 割り込み実験のプログラム

```
#include "3664.h"

#pragma interrupt
void int_irq0(void){
  CLI();
  IRR1 &= 0xfe;
  PDR5 = (PDR5++)&0x3f | (PDR5 & 0xc0);          /* カウントアップ */
  STI();
```

```
 9 }
10
11
12 int main()
13 {
14
15   init_led();                                    /* port5を使うときの初期化 */
16   init_irq0();                                   /* irq0割り込みを使うときの初期化 */
17
18   PDR5 = (0x00 & 0x3f) | (PDR5 & 0xc0);          /* PDR5は, port5のデータレジスタ */
19
20   while(1){
21     SLEEP();
22   }
23 }
```

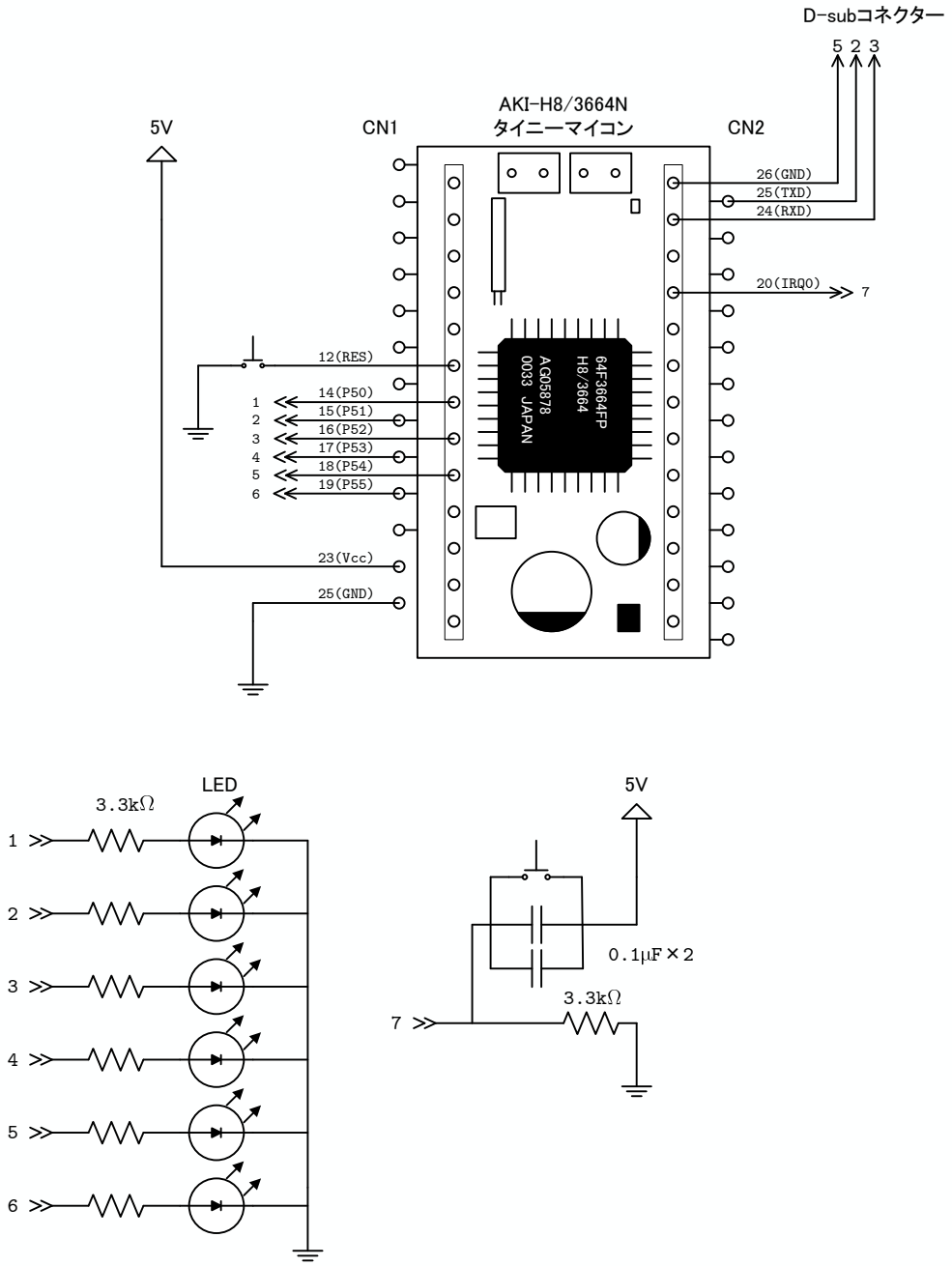

図 11: スイッチによる割り込み (IRQ0) 実験回路

3.5 PWM 実験

パルスの繰り返しやデューティ比—図 12 の H(5V) の時間の割合—を変化させることにより，機器を制御することができる．パルスを変化させて制御することを，PWM 制御 (Pulse Width Modulation) と言う．ここでは，H8 マイコンのレジスター GRA と GRD を変えることにより，PWM 制御の実験を行う．

ここでの実験は，主に文献 [2] を参考にした．

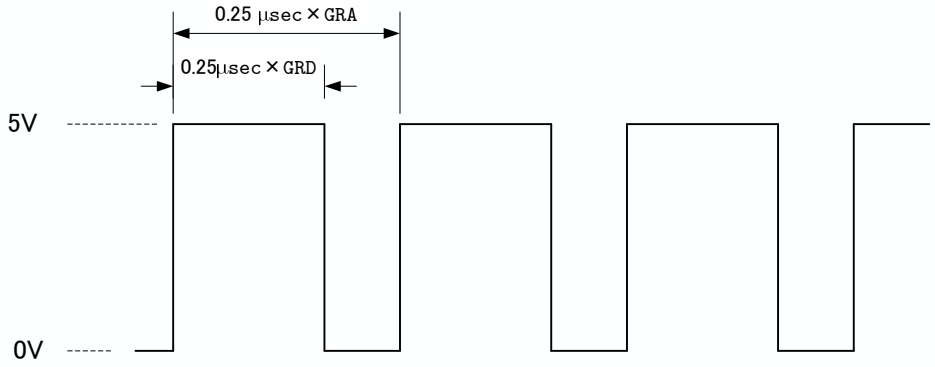

図 12: PWM のパルス幅

3.5.1 トランジスター

ここの実験では，トランジスターをスイッチング素子として使う．実験に先立って，以下の練習問題によりトランジスターの動作内容を理解せよ．

[**練習 8**] インターネットを使って，トランジスターの動作内容を理解せよ．

[**練習 9**] ダーリントン接続を説明せよ．

3.5.2 音を鳴らす

PWM の波形を使って，音を鳴らす．リスト 5 のプログラムでは，IRQ0 に接続されたスイッチを押すごとに，PDR5 の値が増加して，LED のビットがひとつずつ変化—2 進数のビットパターンで+1 増加—-する．それとともに，GRA の周期が 3/4 ずつ短くなる．ただし，デューティ比はいつも 50%である．そのため，音の周波数が，スイッチを押すごとに 4/3 倍と高くなる．

H8 マイコンで作られた PWM 波形は，CN2 の 13 番ピン (FTIOD) から出力される．その出力をトランジスターで増幅し，スピーカーをならしている．

実験順序 以下実験順序を示す．注意事項等については，p.10 の「複数の LED 点灯実験」と同じである．

1. 図 13 のような回路をブレッドボード上に作成する．ただし，まだ電源は接続しない．トランジスターの極性を間違えない こと．
2. 先のプログラム「experiment.c」を書き直して，音を鳴らす実験プログラム「リスト 5」を作成する．
3. コンパイルする．コマンドは,「make」である．
4. make の結果，できあがったプログラムを H8 マイコンへ転送する．
5. H8 マイコンを実行させる．割り込みのスイッチを押す毎に周波数が高くなる．

リスト 5: PWM によるスピーカーを鳴らすプログラム

```
1  #include "3664.h"
2
3  #pragma interrupt
4  void int_irq0(void){
5    CLI();
6    IRR1 &= 0xfe;
7
8    PDR5 = (PDR5++)&0x3f | (PDR5 & 0xc0);          /* カウントアップ */
9
10   GRA=GRA/4*3;                                   /* 音の周期を3/4に */
11   GRD=GRA/2;
12
13   STI();
14 }
15
16
17 int main()
18 {
19
20   init_led();                                    /* port5を使うときの初期化 */
21   init_irq0();                                   /* irq0割り込みを使うときの初期化 */
22   init_pwm();
23
24   PDR5 = (0x00 & 0x3f) | (PDR5 & 0xc0);          /* PDR5は, port5のデータレジスタ */
25
26   GRA=0xffff;                                    /* 音の初期値 */
27   GRD=GRA/2;
28
29   start_pwm();                                   /* pwm 信号スタート */
30
31   while(1){
32     SLEEP();
33   }
34 }
```

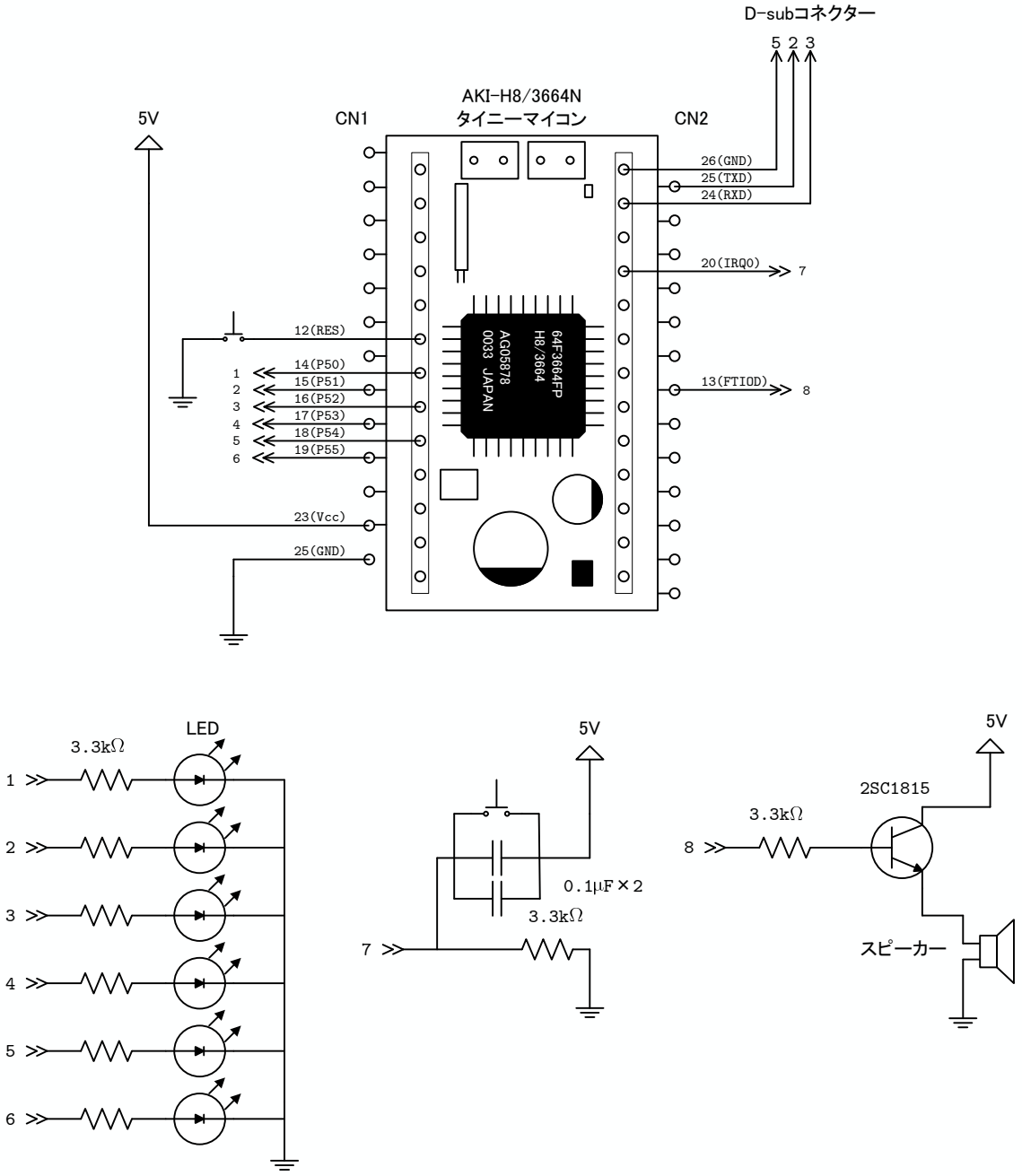

図 13: PWM を使ったスピーカーを鳴らす実験回路

3.5.3 DC モーターの速度制御

PWM を使って，DC モーターの速度制御を行う．DC モーターの回転数は平均電流に，大体，比例する．PWM のデューティ比は，平均電流に比例する．そのため，デューティ比を変化させることにより DC モーターの速度を制御することが可能となる．リスト 6 のプログラムでは，GRD レジスターの値を 1/16 ずつ増加させている．16 回ボタンを押すとデューティ比が 100%となる．

H8 マイコンで作られた PWM 波形は，CN2 の 13 番ピン (FTIOD) から出力される．その出力をダーリントン接続したトランジスターで増幅し，モーターを回している．

実験順序　以下実験順序を示す．注意事項等については，p.10 の「複数の LED 点灯実験」と同じである．

1. 図 14 のような回路をブレッドボード上に作成する．ただし，まだ電源は接続しない．トランジスターの極性を間違えないこと．
2. 先のプログラム「experiment.c」を書き直して，DC モーターの速度制御の実験プログラム「リスト 6」を作成する．
3. コンパイルする．コマンドは,「make」である．
4. make の結果，できあがったプログラムを H8 マイコンへ転送する．
5. H8 マイコンを実行させる．割り込みのスイッチを押す毎にモーターの回転数が増える．

リスト 6: モーターの速度制御のプログラム

```
1  #include "3664.h"
2
3  #pragma interrupt
4  void int_irq0(void){
5    CLI();
6    IRR1 &= 0xfe;
7
8    PDR5 = (PDR5++)&0x3f | (PDR5 & 0xc0);          /* カウントアップ */
9
10   GRD+=0x1000;                                    /* PWMのHの幅を1/16ずつひろげる */
11
12   STI();
13 }
14
15
16 int main()
17 {
18
19   init_led();                                  /* port5を使うときの初期化 */
20   init_irq0();                                 /* irq0割り込みを使うときの初期化 */
21   init_pwm();
22
23   PDR5 = (0x00 & 0x3f) | (PDR5 & 0xc0);        /* PDR5は，port5のデータレジスタ */
24
25   GRA=0xffff;                                  /* モーターの初期速度 */
26   GRD=0x0000;
27
28   start_pwm();                                 /* pwm 信号スタート */
29
30   while(1){
31     SLEEP();
32   }
33 }
```

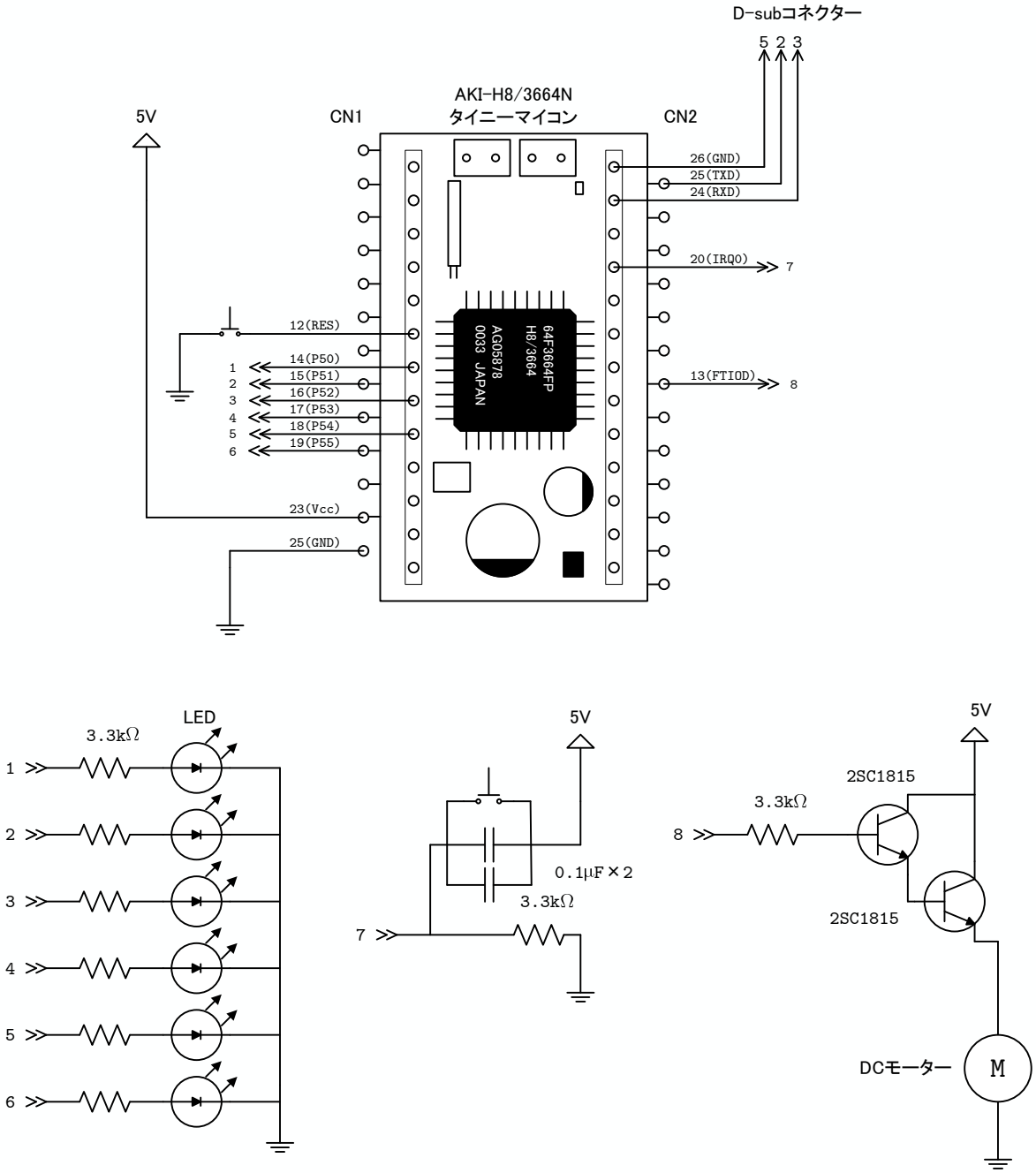

図 14: PWM による DC モーター動作実験回路

3.6 LCD 表示実験

ここでは，LCD(液晶ディスプレイ:Liquid Crystal Display) に文字を表示させる実験を行う．C 言語で書かれた H8 マイコンのプログラムにより，LCD をコントロールして任意の文字を表示させる．プログラムおよびハードウェアーの設定は，主に文献 [3] を参考にした．

3.6.1 LCD

液晶は電圧を加えることにより，光のシャッターとして使うことができる．これを利用したものが LCD である．ここの実験では，16 文字 ×2 行の表示ができる LCD を使う．この LCD はひとつの文字を 7×5 ドットで構成しているので，合計 1120 ドットある．すなわち，1120 個の小さい点を液晶により ON/OFF することにより，いろいろな文字を表示しているのである．

図 15 にこの実験で使う LCD の外観とピン番号を示す．ここで使う LCD はデータを 4 ビットと 8 ビットのいずれかで転送することができる．今回の実験では，接続する線の数を減らすために，4 ビットモードで使用する．

図 15: LCD の外観とピン番号

3.6.2 ポテンションメーター

回路基板に取り付ける部品で，抵抗が可変のものを可変抵抗といったり，トリマーあるいはポテンションメーターと言う．ここでも，ポテンションメーターを使うので，簡単に原理を説明する．そして，実際にポテンションメーターの動作を実験で確かめる．

図 16 に示すように，それには 3 本の電極がある．2 本は抵抗の両端に接続されており，残りの 1 本は抵抗の途中に接続されている．そのため図 16 のように接続することにより，0〜5[V] の電圧を取り出すことができる．

ここでは，10[kΩ] のポテンションメーターを使って，LCD のコントラストの調整に用いている．LCD のコントラストは，3 番ピンの電圧で決まる．その電圧をポテンションメーターで供給する．

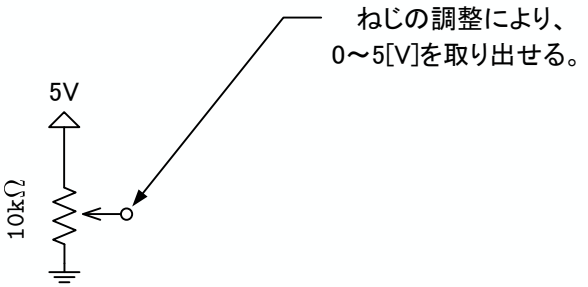

図 16: ポテンションメーター

[**練習 10**] ポテンションメーターの抵抗が図 16 のようになっていることをテスターで確認せよ．

[**練習 11**] 図 16 の回路を作成し，0〜5[V] が取り出せることを確認せよ．

3.6.3 H8 による LCD 制御

H8 に接続した LCD に文字を出力する実験を行う．H8 マイコンの Port5 出力 (CN1 の 14–19 番) の 6 本の線を使って，LCD の表示を制御する．このうち 4 本 (CN1 の 14–17 番) がデータ線で，残りの 2 本が制御線である．

実験順序 以下実験順序を示す．注意事項等については，p.10 の「複数の LED 点灯実験」と同じである．

1. 図 17 のように回路をブレッドボード上に作成する．ただし，電源は，接続しない．
2. 先のプログラム「experiment.c」を書き直して，LCD 制御の実験プログラム「リスト 7」を作成する．
3. コンパイルする．コマンドは,「`make`」である．
4. `make` の結果，できあがったプログラムを H8 マイコンへ転送する．
5. H8 マイコンを実行させる．文字が表示されるはずである．

LCD を制御するために，以下のような関数を用意した．必要に応じて使うこと．これらの関数のソースプログラムは，ファイル h8c.c にあるので興味のある者は見るとよいだろう．

関数 `void LCD_init(void)`
役割 LCD を使うための初期化を行う．LCD を使うときに，最初に必ず呼び出さなくてはならない．

関数 `void LCD_clear(void)`
役割 LCD の画面の文字を全て消去する．

関数 `void LCD_locate(int row, int col)`
引数 第一引数:行数　第二引数:列数
役割 引数で指定した位置にカーソルを移動させる．

関数 `void LCD_putstr(char *string)`
引数 文字列を示すポインター
役割 文字列を LCD に表示させる．

関数 `void LCD_putchar(char c)`
引数 表示させたい文字 (1 文字)
役割 文字を LCD に表示させる．

関数 `void LCD_putint(char i)`
引数 表示させたい整数．
役割 整数を LCD に表示させる．

関数　`void LCD_putfloat(double x, int d)`
引数　第一引数:表示させたい倍精度実数　第二引数:小数点以下の桁数
役割　倍精度実数を浮動小数点数として，指定の桁数で表示する．

関数　`void LCD_putbit8(unsigned char d)`
引数　表示させたい整数 (1 バイト)
役割　1 バイトのビットパターンを表示させる．

関数　`void wait_ms(int s)`
引数　ミリ秒
役割　指定の時間，CPU の動作を止める．

リスト 7: Hello world!を表示するプログラム．

```
1  #include <stdlib.h>
2  #include "3664.h"
3  #include "h8c.h"
4
5
6  int main()
7  {
8
9    LCD_init();                          // LCDの初期化
10   LCD_clear();                         // LCDの画面クリアー
11   LCD_locate(1,1);                     // カーソル位置変更
12   LCD_putstr("Hello World !.");        // 文字列出力
13   LCD_locate(2,1);
14   LCD_putstr("From Akita.");
15
16   while(1){
17     SLEEP();
18   }
19 }
```

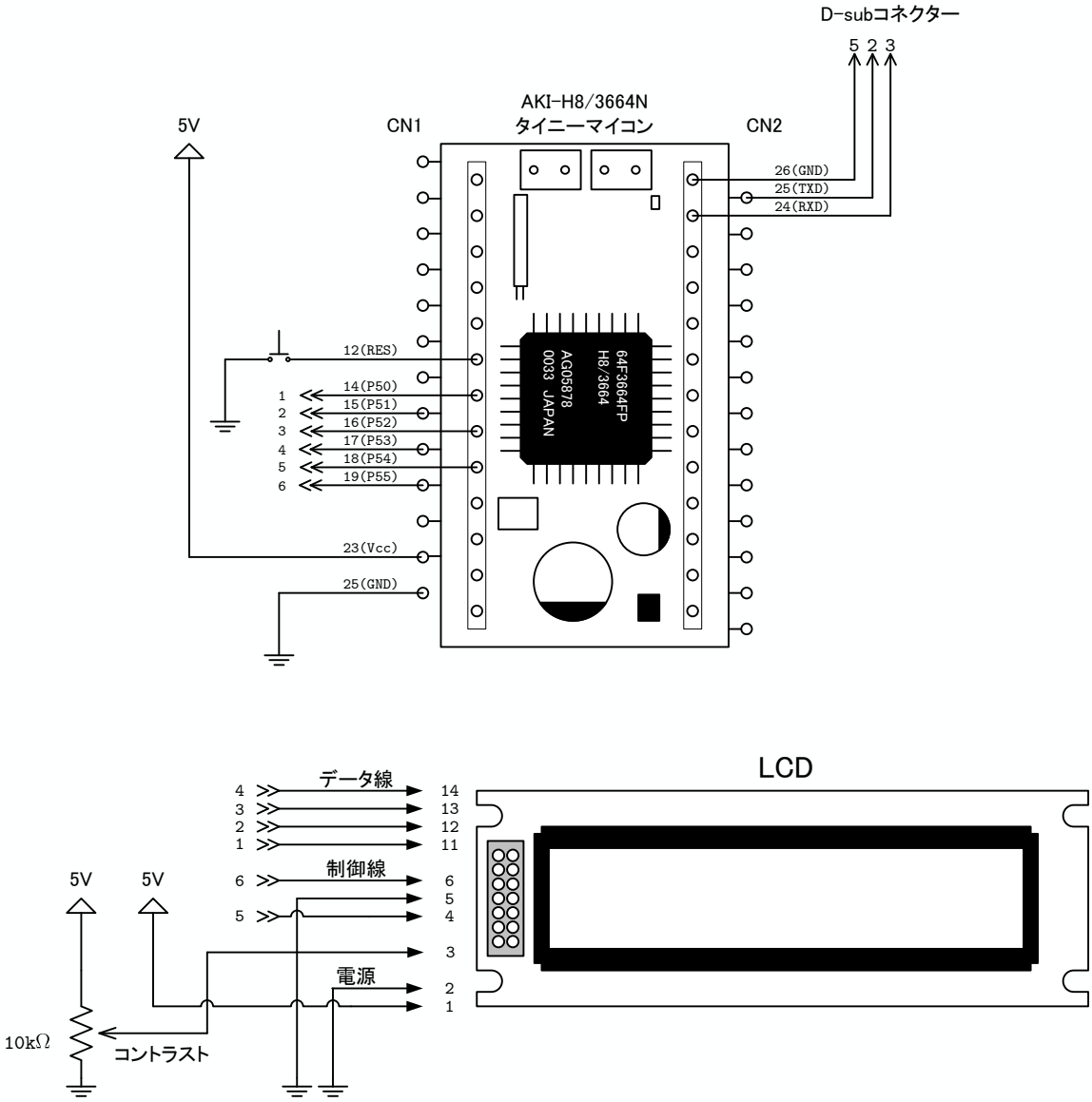

図 17: H8 による LCD 制御実験の回路

[**練習 12**] リスト 7 を書き換えて，自分の名前を表示 (1 行目:姓，2 行目:名) せよ．

[**練習 13**] リスト 7 を書き換えて，適当な整数を表示せよ．ただし，`LCD_putint()` 関数を使うこと．

[**練習 14**] リスト 7 を書き換えて，適当な実数を表示せよ．ただし，`LCD_putfloat()` 関数を使うこと．

3.7 A/D 変換器実験

ここでは，H8 マイコンの AD 変換器 (Analog to Digital Converter) の実験を行う．

3.7.1 電圧計

H8 マイコンでは，アナログ信号を取り込んで処理することができる．アナログ信号をコンピューターで処理するためには，デジタル情報に変換する必要がある．その変換を行う部分を AD 変換器と言う．H8 マイコンには AD 変換を行う回路が組み込まれている．ここでは，それを使って，デジタル電圧計を作成する．

H8 マイコンは，CN1 の 23 番ピンに印加されている電圧 (大体 5 [V]) を 10 ビットでデジタルに変換する．10 ビットで表現できる最大値は 1111111111=1023 である．したがって，5/1023=4.88 [mV] が分解能である．

H8 マイコンには 8 個のアナログ入力端子があるが，ここでは CN1 の 10 番ピンの AN0 のみ使う．この端子に印加する電圧は，AD 変換を行う際の比較電圧 (CN1 の 23 番ピン) よりも小さくしなくてはならない．また，逆電圧も印加してはならない．回路が破損するからである．そのため，ここでの実験ではダイオードを用いた保護回路つける．

実験順序　以下実験順序を示す．注意事項等については，p.10 の「複数の LED 点灯実験」と同じである．

1. 図 18 のように回路をブレッドボード上に作成する．ただし，電源は，接続しない．
2. 先のプログラム「experiment.c」を書き直して，電圧計実験プログラム「リスト 8」を作成する．
3. コンパイルする．コマンドは,「`make`」である．
4. `make` の結果，できあがったプログラムを H8 マイコンへ転送する．
5. H8 マイコンを実行させる．すると，電圧が表示される．
6. 1[kΩ] のポテンションのネジを回して，電圧を変化させてみよ．

AD 変換器を制御するために，以下のような関数を用意した．必要に応じて使うこと．これらの関数のソースプログラムは，ファイル h8c.c にあるので興味のある者は見るとよいだろう．

関数　`void ADC_init(void)`
役割　AD 変換器を使うための初期化を行う．AD 変換器を使うときに，最初に必ず呼び出さなくてはならない．

関数　`int ADC(void)`
役割　AN0(CN1 の 10 番ピン) に接続さたアナログ電圧よ読み取り，戻り値として返す．戻り値は 0〜1023 の整数値である．0 が 0 [V] で，1023 が CN1 の 23 番ピンの電圧となる．

リスト 8: 電圧を測定するプログラム

```
1  #include <stdlib.h>
2  #include "3664.h"
3  #include "h8c.h"
4
5
6  int main()
7  {
8    int vdata;
9    double v;
10
11   LCD_init();
12   LCD_clear();
```

```
13    LCD_locate(1,1);
14    LCD_putstr("Start ADC");
15    wait_ms(1000);
16    LCD_clear();
17
18    ADC_init();
19
20    while(1){
21      vdata= ADC();
22
23      v = (double)vdata/1023*5.1;
24
25      LCD_clear();
26      LCD_locate(1,1);
27      LCD_putstr("ADC result");
28      LCD_locate(2,1);
29      LCD_putfloat(v,3);
30      LCD_locate(2,10);
31      LCD_putstr("[V]");
32
33      wait_ms(100);
34    }
35  }
```

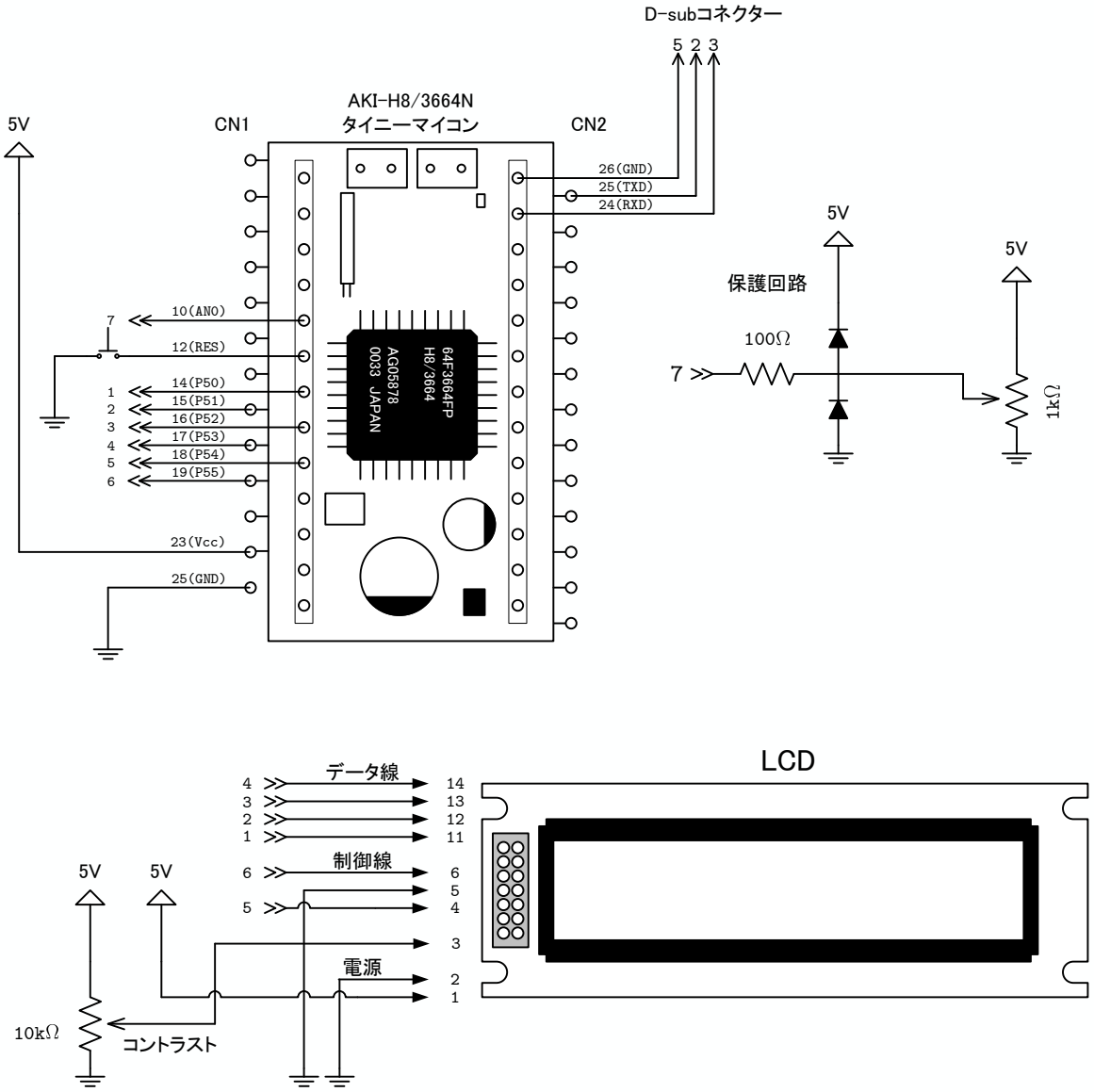

図 18: A/D 変換器を使った電圧計の回路

[練習 15] リスト 8 のプログラムだとノイズによる電圧の揺らぎが大きい．複数回測定の平均値を祖テク位電圧することにより，この問題は解決できる．10000 回測定して，その平均電圧を表示するプログラムに直せ．測定精度は，どの程度向上したか?

3.7.2 温度計

電圧計を応用して温度計を作製する．

温度センサー 温度センサーは，ナショナルセミコンダクター社の LM35 を使う．これは図 19 のような形状をしている．G をグラウンド電位 (0 [V]) にして, +Vs に 4〜20 [V] を印加すると，Vout に 10 [mV/℃] が

出力される．20 [℃] だと，200 [mV] が出力される．

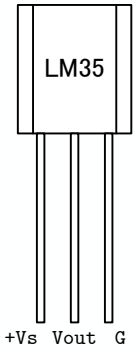

図 19: 温度センサー

[**練習 16**] ブレッドボード上で，LM35 の G をラウンドウンド電位に，+Vs に 5 [V] を印加して，Vout を測定せよ．現在の室温と比べてどうか?

OP アンプ　温度センサー LM35 の出力は，H8 マイコンの AD 変換器の測定レンジに比べ小さい．20[℃] の場合のセンサー出力 0.2 [V] に対して，AD 変換器は 5 [V] まで入力できる．センサーの出力を増幅させて，測定精度の向上をはかる．

電圧の増幅には OP アンプ (operational amplifier) をつかう．OP アンプ使った信号の増幅は簡単で，広範囲に使われている．ここでは，非反転増幅という回路をつかう．その回路は，図 20 のようにする．このようにすると入力電圧 (V_{in}) と出力電圧 (V_{out}) は，

$$V_{out} = \frac{R_s + R_f}{Rs} V_{in} \tag{1}$$

となる．$(R_s + R_f)/R_s$ が増幅率になる．抵抗の組み合わせで，簡単に増幅率を変えることができる．

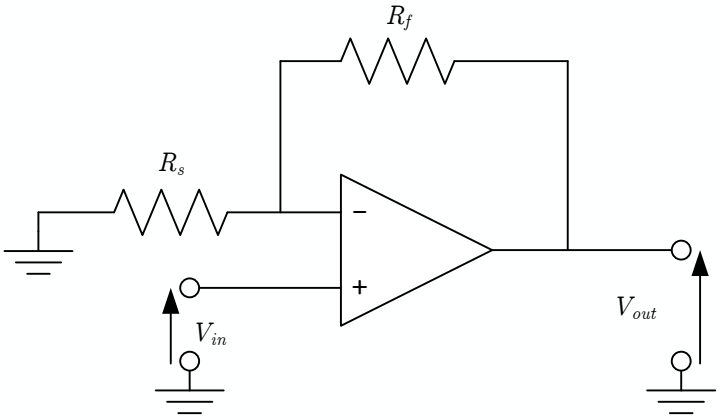

図 20: OP アンプを使った非反転増幅回路

この実験で使う OP アンプはナショナルセミコンダクター社の LMC662 で，図 21 に示すように，ひとつのパッケージに 2 つの OP アンプが内蔵されている．

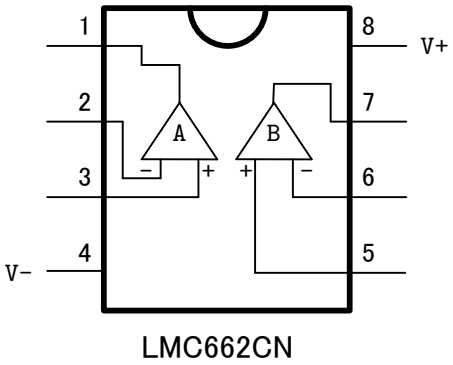

図 21: OP アンプ (LMC662CN) の内部

[**練習 17**] 図 22 の回路の増幅率が 11 倍になっていることを式で示せ．

温度計の作製

実験内容 電圧計の回路および温度センサー，OP アンプを使って温度計を作成する．

実験順序 以下実験順序を示す．注意事項等については，p.10 の「複数の LED 点灯実験」と同じである．

1. 図 22 のように回路をブレッドボード上に作成する．ただし，電源は接続しない．

2. 先のプログラム「experiment.c」を書き直して，温度計の作成実験のプログラム「リスト 9」を作成する．

3. コンパイルする．コマンドは,「`make`」である．

4. `make` の結果，できあがったプログラムを H8 マイコンへ転送する．

5. H8 マイコンを実行させる．すると，温度が表示される．

   - ブレッドボードから，5 [V] の電源を取り外す．
   - H8 マイコン基板の JP2 と JP3 のジャンパーピンを取り外し，オープンにする．
   - ブレッドボードに，5 [V] を供給する．
   - プログラムで指定した通りに表示される．
   - プログラムを最初から実行させたければ，RES スイッチを押す．

6. 温度センサーを触ったりして，温度の変化を調べよ．

リスト 9: 温度を測定するプログラム

```
#include <stdlib.h>
#include "3664.h"
#include "h8c.h"


int main()
{
  int i, sum_ad, N=10000;
  double t;

  LCD_init();
  LCD_clear();
  LCD_locate(1,1);
  LCD_putstr("Start ADC");
  wait_ms(1000);
  LCD_clear();

  ADC_init();

  while(1){
    sum_ad=0;

    for(i=0;i<N;i++){
      sum_ad += ADC();
    }

    t = (double)sum_ad/N*5.1/1023/11.0/0.01;

    LCD_clear();
    LCD_locate(1,1);
    LCD_putstr("Temperature");
    LCD_locate(2,1);
    LCD_putfloat(t,4);
    LCD_locate(2,11);
    LCD_putstr("[deg]");
  }
}
```

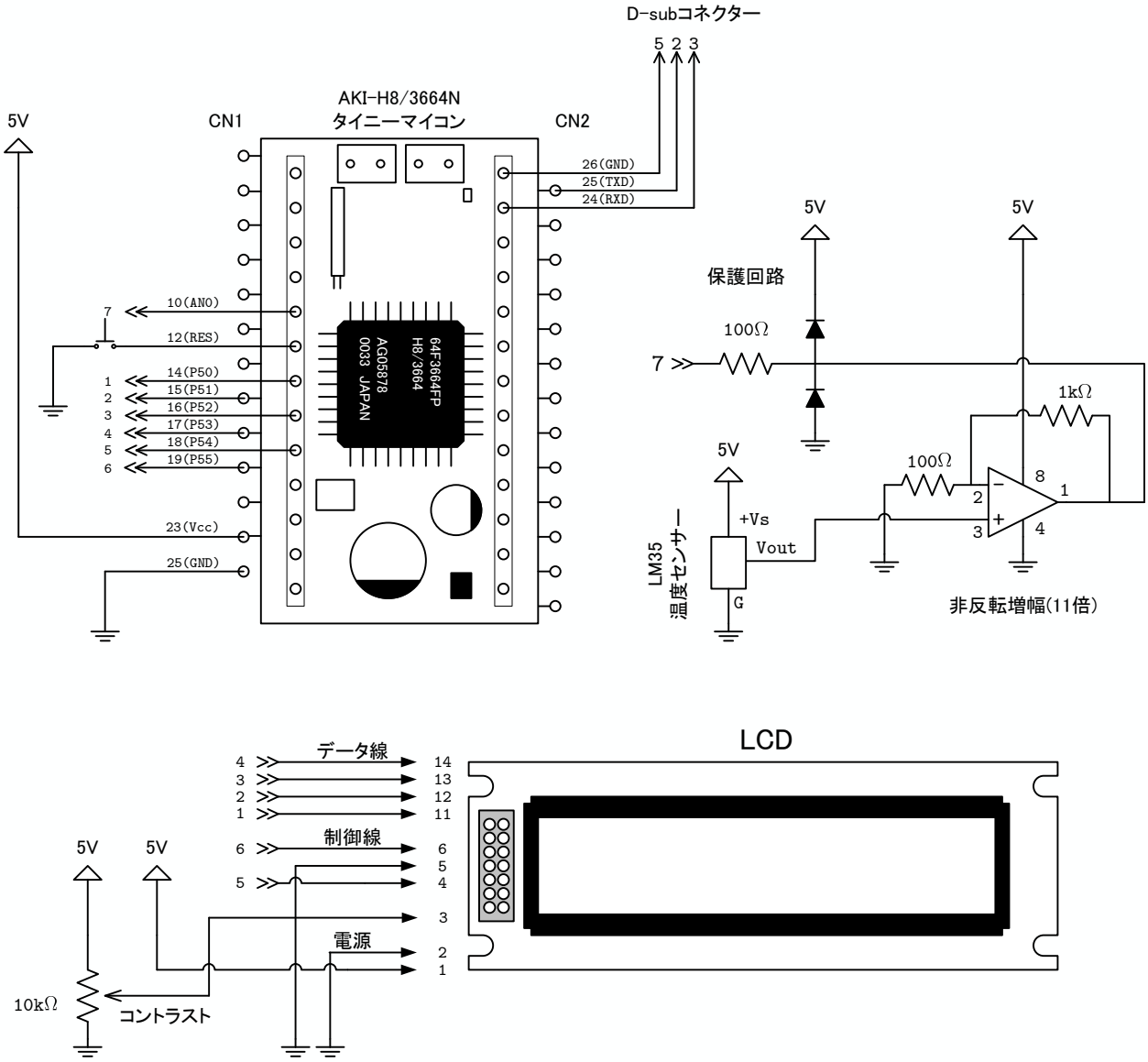

図 22: A/D 変換器を使った温度計の回路

4 考察課題

1. コンピューターの基本構成要素を述べよ.
2. マイコンが使われている身近な機器にはどんなものがあるか?
3. 家電にような民生品には，マイコンではなく，他の半導体部品で制御しているものもある．どのような半導体部品を使っているか，調べよ.
4. H8 以外のマイコンにはどのような物があるか．H8 との違いを，表にまとめよ.

5. H8 マイコンはどのようなことに使えるか考えよう．高専祭のクラス展示に応用するとして，どのような装置を作るとうけるだろうか．

5　レポートのまとめかたの注意

結果について　この実験では，レポートの「結果」の節の書き方が，今までのレポートと異なる．これまでの実験では,「結果」には測定結果をまとめたグラフや表を書いていた．ここでは，それに代わるものとして，各実験での回路の動作内容を記述する．さらに，練習問題の解答も記述すること．

デジタル回路の実験では，グラフや表にまとめられないものが多い．それでも，レポートには，実験方法に示した実験の結果を記述する必要がある．実験がどうなったか—ということを報告しなくてはならない．

考察について　これは，通常の実験と同じである．実験に対して，どのように考えるか—を実験を行った者の視点で考える．

参考文献

[1] 鹿取佑二. C 言語で H8 マイコンを使いこなす. (株) オーム社, 2005. ソフトウェアーの作成に大いに参考にした．自分でプログラムを作成する場合，揃えたい本である．

[2] 島田義人. H8/Tiny マイコン完璧マニュアル. CQ 出版社, 2005. 細部に渡って説明がある．ここでの実験では PWM 制御について参考にした．

[3] 米田聡. はじめる　組込み Linux. ソフトバンククリエイティブ (株), 2007. H8 を組込みシステムで使おう—という趣旨の本．かなり詳しく書いているので，H8 を使うならば，参考書として揃えたい本である．ここでの実験では LCD の制御方法について，参考にした．