

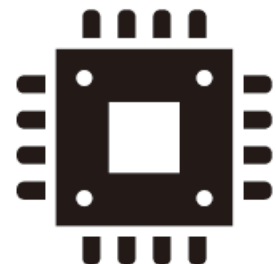
C 言語ではじめる Raspberry Pi 徹底入門



Chapter 13

7セグメント LED 表示器で 数字を表示する

本章では7セグメント LED 表示器を使用して、数字を表示します。多様な表示が可能な LCD が普及していますが、数字だけの表示なら大きく明るい7セグメント LED 表示器が使用されています。7セグメント LED 表示器の構造、ダイナミック点灯方式の原理、時刻を表示させる課題を学びます。また、トランジスタの概要やスイッチング回路についても学びます。



目 次

Chapter13	7セグメント LED 表示器で数字を表示させる	1
はじめに		3
13.1	本章で準備するもの	4
13.2	7セグメント LED 表示器とは	5
13.3	数字を表示させる	8
	回路図／配線図／フローチャート／ソースコード／動作確認	
13.4	トランジスタとは	15
13.5	4桁の数字をダイナミック点灯させる	18
	4桁7セグメント LED 表示器 OSL40562-LR の構造／ダイナミック点灯方式とは	
	／回路図／配線図／フローチャート／ソースコード／動作確認	
13.6	章末問題	29
参考文献		32

必ずお読みください

本書では、次のようなシステム構成を前提に解説しています。

- 本体 : Raspberry Pi 3/3B+および4B
- OS : Raspbian (Buster) 2019-09-30版 (NOOBS Ver.3.2.1)
- ディスプレイ : HDMI 仕様
- キーボード・マウス : USB 仕様
- SD カード : microSD SDHC 16GB スピードクラス10
- ネットワーク環境 : Raspberry Piに内蔵した無線LAN (Wi-Fi) でインターネットに接続

本書は、Windows PCやインターネットの一般的な操作を一通りできる方を対象にしているため、基本的な操作方法などは解説していません。C言語や電子回路の基本的な知識についても、習得されていることを前提にしています。電子部品や工具などを使用する際は、ケガなどをされないようご注意ください。また、電子部品は静電気に弱いので取り扱いにもご注意ください。

本書のサポートページについて

本書で利用するソースコードなどが記述されたサンプルファイルと追加情報や、正誤表、本書発行後にバージョンアップなどにより設定手順などに変更が生じた場合の補足情報は、本書サポートページに掲載しています。

- 本書サポートページ
<https://gihyo.jp/book/2020/978-4-297-11299-8>

本書に記載された内容は、情報の提供のみを目的としています。したがって、本書を用いたご利用は、必ずお客様自身の責任と判断によって行ってください。これらの情報によるご利用の結果について、技術評論社および著者はいかなる責任も負いません。

本書記載の情報は、2020年1月現在のものを掲載していますので、ご利用時には変更されている場合もあります。また、ソフトウェアに関する記述は、とくに断わりのないかぎり、2020年1月時点での最新バージョンをもとにしています。それぞれバージョンアップされる場合があり、本書での説明とは内容などが異なってしまうこともあり得ます。本書ご購入の前に、必ずバージョン番号などをご確認ください。本書の配線図は「Fritzing」(<https://fritzing.org/>)を使って描いたものを使用しています。

以上の注意事項をご承諾いただいたうえで、本書をご利用願います。これらの注意事項をお読みいただかずに、お問い合わせいただいても、技術評論社および著者は対処しかねます。あらかじめ、ご承知おきください。

本書で記載されている会社名、団体名、製品名、サービス名などは、一般に関係各社・団体の商標または登録商標です。なお、本書では™、®などは明記していません。また、本書では、通称やそのほかの名称で記述する場合がありますが、あらかじめご了承ください。

はじめに

「Chapter13 7 セグメント LED 表示器で数字を表示させる」は、「C 言語ではじめる Raspberry Pi 徹底入門」（以下、「徹底入門」）の追加の章として作成しました。

多様な表示が可能な LCD が普及していますが、数字だけの表示なら大きく明るい 7 セグメント LED 表示器が使用されています。本章では 7 セグメント LED 表示器の構造、4 桁の表示器を使用して、ダイナミック点灯方式の原理、時刻を表示させる課題を学びます。また、トランジスタの概要やスイッチング回路についても学びます。

本章の学習の順番は、「徹底入門」の Chapter4 の LED への信号の出力（GPIO の出力を使う）を学んだ後に学習することをおすすめします。

なお、本章で組み立てる 7 セグメント LED 表示器の回路の信号線が、「徹底入門」と重複しますので、新しいブレッドボードに配線してください。

Chapter13 のソースコードは、著者の GitHub の Web サイトからダウンロードできます。

<https://tamamori2020.github.io/raspi-gh1/>

13.1. 本章で準備するもの

本章の電子工作で新たに必要とする主なものを表 13-1 に示します。参考品の欄に初心者の方がイメージしやすいように写真と入手情報等を載せましたが、持ち合わせの部品や相当品で結構です。なお、「徹底入門」で準備したものも使用します。

表 13-1 本章で準備する主なもの

名 称	個数	備 考	参 考 品 (秋月電子通商)
7セグメント LED 表示器	1	カソードコモンタイプを使用します。	I-00640 (①)
4 桁 7セグメント LED 表示器	1	カソードコモンタイプを使用します。	I-03673 (②)
トランジスタ 2CS1815-Y	4	4 桁 7セグメント LED 表示器の点灯回路に使用。 Y クラスを使用しますが、GR, BL クラスでも可。	I-04268 (③)
抵抗 100Ω	8	4 桁 7セグメント LED 表示器の点灯回路に使用。	R-25101
抵抗 390Ω	8	1 桁 7セグメント LED 表示器の点灯回路に使用。	R-25391
抵抗 4.7kΩ	4	4 桁 7セグメント LED 表示器の点灯回路に使用。	R-25472



13.2. 7セグメント LED 表示器とは

7 (ナナ)セグメント LED 表示器¹は、主にアラビア数字を表示するための電子部品で、デジタル時計、メーターなどの電子機器、エレベーターの表示などに使用されています。数字を表現するのに LED を 7 個 (セグメント) 使用することから、7セグメント LED 表示器と呼ばれています²。

表示面 (図 13-1(a)) には、いわゆるデジタル数字がデザインされ、底面 (図 13-1(b)) を見ると、LED が実装されたリードフレームがランプハウスと呼ばれるケースに挿入されて、エポキシ樹脂で封止されています。

図 13-2 に示すように、デジタル数字の各セグメントには a から g までの名称が付けられています。dp は decimal point の略語で小数点のことです。dp を含めると LED の数は 8 個になります。数字 0 から 9 とアルファベット A から F の 16 進数の表示パターンを図 13-3 に示します。なお、B と D は小文字になります。

図 13-1 7セグメント LED 表示器の外観

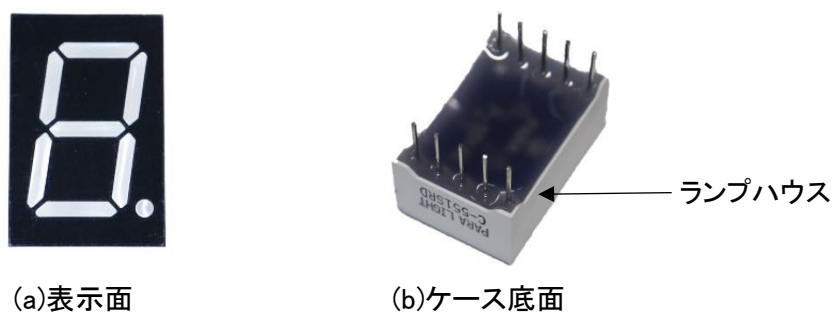
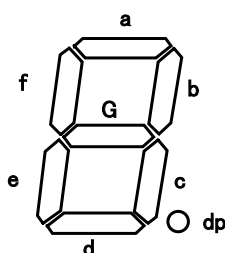


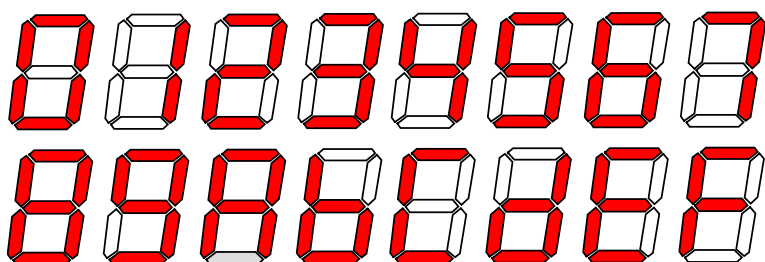
図 13-2 デジタル数字とセグメントの名称



¹ 通称「ナナセグ」と言う。

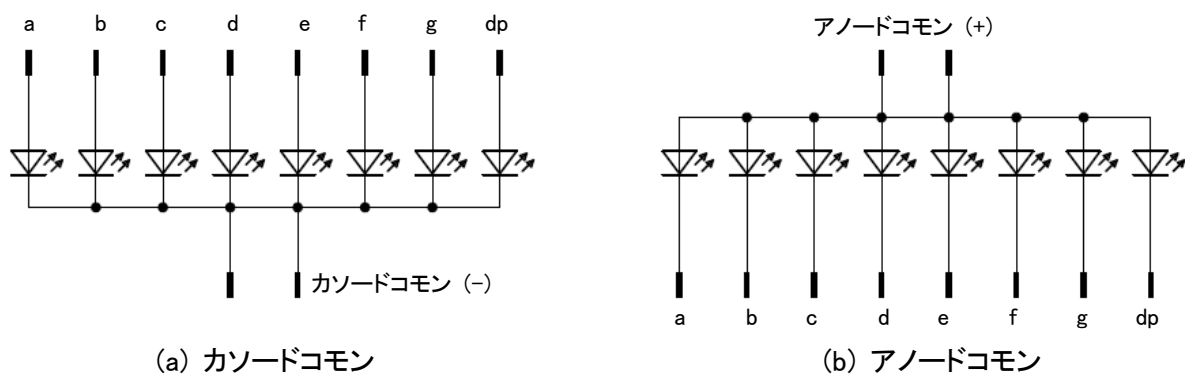
² 対角線と中央を縦に通る線を追加し、横線を 2 つに分割した 14 セグメントや 16 セグメントの LED 表示器は、英字を表示することが出来る。

図 13-3 16進数の表示パターン



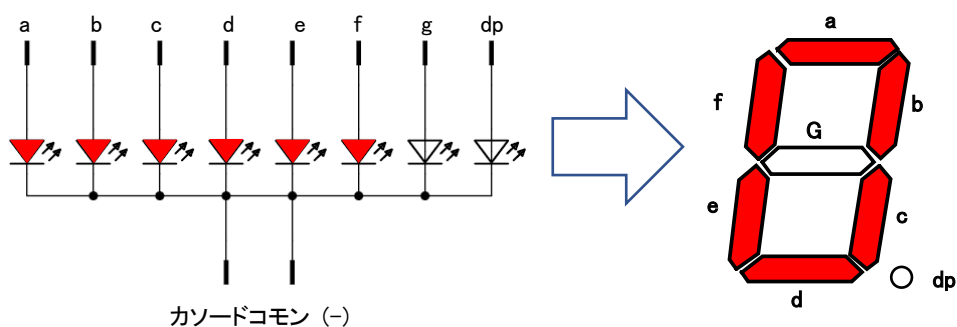
7セグメント LED 表示器の内部回路には、**アノードコモン**と**カソードコモン**の 2 種類があります(図 13-4)。コモンは共通の意味です。カソードコモンは各 LED のカソード側が結線され、アノードコモンはアノード側が結線されています。アノードコモンとカソードコモンでは表示回路の設計は異なります。コモンの種類を間違えると点灯しないので、購入するときは注意してください。

図 13-4 カソードコモンとアノードコモンの 7 セグメント LED 表示器の内部配線



【例】カソードコモンで 0 を表示させる場合、セグメント a から f に電圧を加えると 6 個の LED が発光して、0 に見えます(図 13-5)。

図 13-5 0 の表示パターンの表示例



BCD(Binary-Coded Decimal:2 進化 10 進数)³を入力して、セグメント a から g の表示パターンを出力する回路を 7 セグメント・デコーダと呼びます。7 セグメント・デコーダの真理値表を表 13-2 に示します。カソードコモンの 7 セグメント LED 表示器用に、74LS47 や 74HC4511 などの専用 IC があります。本章では、7 セグメント・デコーダ回路をソフトウェアで処理をします。

表 13-2 7 セグメント・デコーダの真理値表

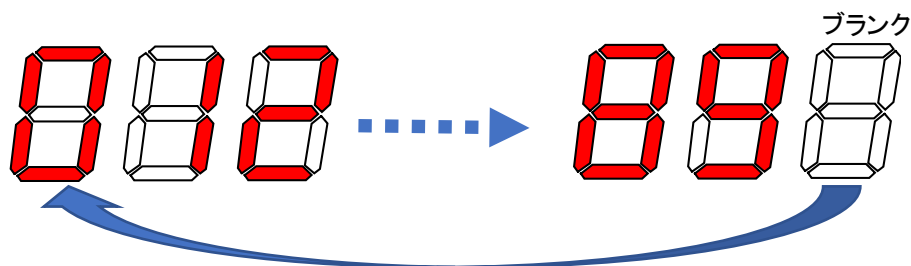
BCD 入力				出力							表示パターン
D	C	B	A	a	b	c	d	e	f	g	
L	L	L	L	H	H	H	H	H	H	L	0
L	L	L	H	L	H	H	L	L	L	L	1
L	L	H	L	H	H	L	H	H	L	H	2
L	L	H	H	H	H	H	H	L	L	H	3
L	H	L	L	L	H	H	L	L	H	H	4
L	H	L	H	H	L	H	H	L	H	H	5
L	H	H	L	L	L	H	H	H	H	H	6
L	H	H	H	H	H	H	L	L	L	L	7
H	L	L	L	H	H	H	H	H	H	H	8
H	L	L	H	H	H	H	L	L	H	H	9

³ BCD は 0 から 9 の 10 進数を、0000 から 1001 の 4bit の 2 進数で表現したもの。2 進数の各 bit の重みが 8-4-2-1 であることから、8421 符号と呼ばれる。なお、1010 から 1111 は使用しない。

13.3. 数字を表示させる

7セグメントLED表示器を使用して、図 13-6 に示すように「0」から順番に1秒毎にカウントアップして表示して「blank」になったら、また、「0」から繰り返し表示するプログラムを作成します。blank (blank) とは、非点灯のことです。

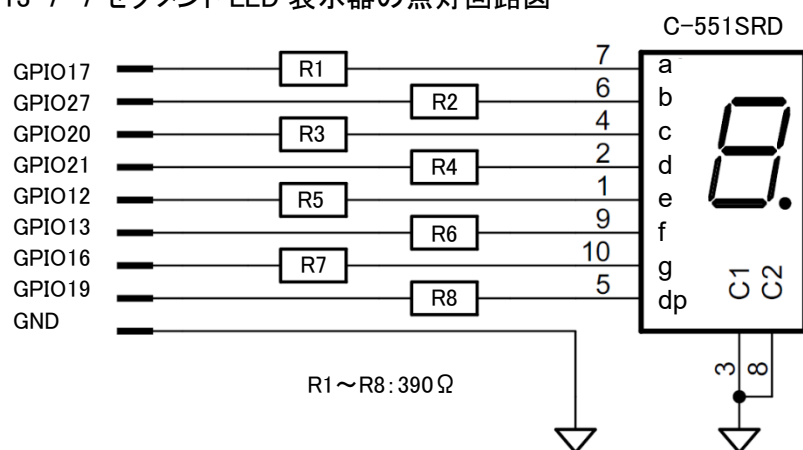
図 13-6 7セグメントLED表示器でカウントアップを繰り返す動作



13.3.1. 回路図

7セグメントLED表示器の点灯回路を図 13-7 に示します。カソードコモンタイプの7セグメントLED表示器 C-551SRD (秋月電子通商: I-00640) を使用します。C-551SRD の C1 (3 番ピン) と C2 (8 番ピン) はカソードコモンです。Chapter4 のLEDの点灯回路 (図 4-6) と同じく順方向電流 I_F を 3mA として、R1 から R8 の抵抗値を 390Ω としました。GPIO 信号の出力が HIGH のとき各 LED は点灯し、LOW のときは消灯します。

図 13-7 7セグメントLED表示器の点灯回路図



13.3.2. 配線図

7セグメント LED 表示器 C-551SRD のピン配列を図 13-8 に示します。配線図は図 13-9 になります。なお、本回路は、「徹底入門」で製作したブレッドボードとは別に、新しいブレッドボードに組み立てます。

図 13-8 表示面から見た C-551SRD のピン配列

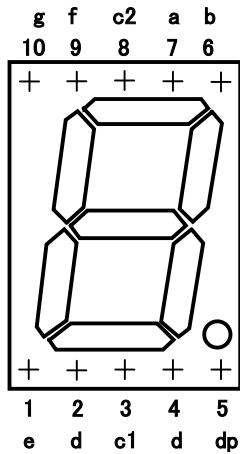
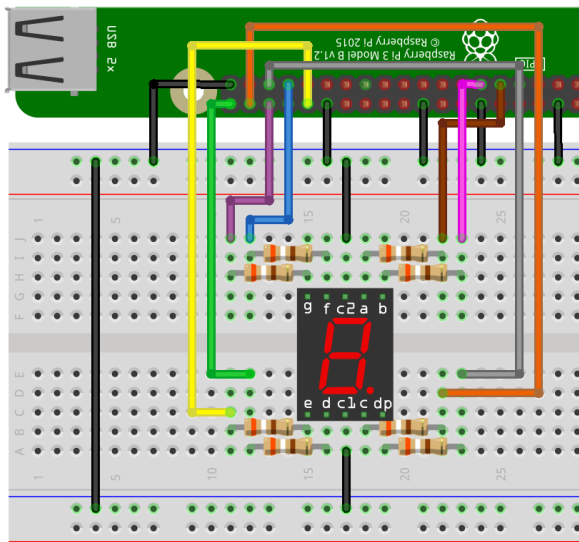


図 13-9 7セグメント LED 表示器の配線図



13.3.3. フローチャート

反復構造のフローチャートは、終了条件を満たすまで同じ処理を繰り返すように、記号「判断」(図 13-10(a))で作成することができますが、「ループ端」(図 13-10(b))を使用すると見やすくなります。反復させる処理をループの始端と終端で挟むだけで表記できます。始端または終端に、初期化、増分、終了条件を表記します。また、始端と終端に同じループ名を記入します。ここでは、ループ端を使用して、フローチャートを作成します。

図 13-10 フローチャートの判断とループ端を使用した反復構造

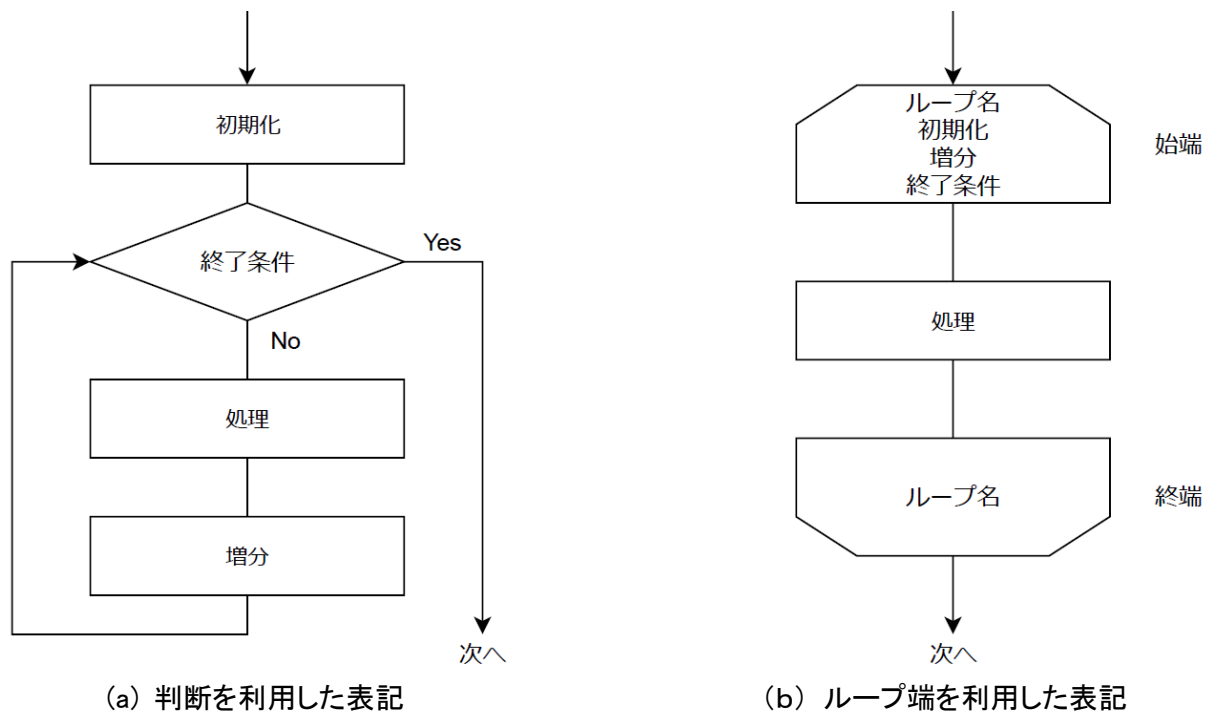


図 13-6 に示したように、7 セグメント LED 表示器に、「0」から順番に 1 秒毎にカウントアップして表示して「blank」になったら、また、「0」から繰り返し表示するプログラムを作成します。フローチャートを図 13-11 に示します。

図 13-11 1桁の数字を表示するフローチャート

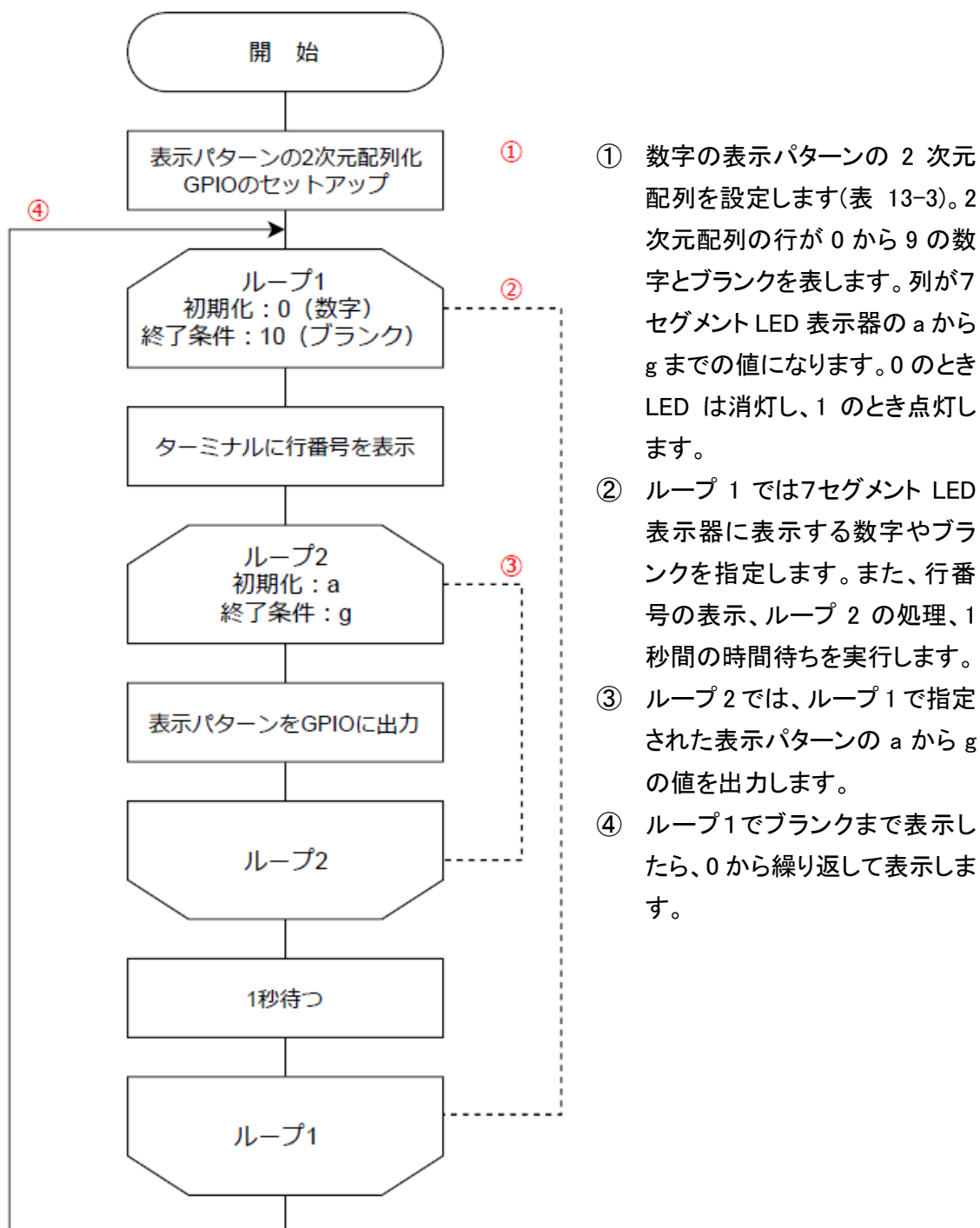


表 13-3 7セグメントLED 表示器用表示パターンの2次元配列

				セグメント名						
				a	b	c	d	e	f	g
				列番号						
				0	1	2	3	4	5	6
表示パターン	0	行番号	0	1	1	1	1	1	1	0
	1		1	0	1	1	0	0	0	0
	2		2	1	1	0	1	1	0	1
	3		3	1	1	1	1	0	0	1
	4		4	0	1	1	0	0	1	1
	5		5	1	0	1	1	0	1	1
	6		6	1	0	1	1	1	1	1
	7		7	1	1	1	0	0	0	0
	8		8	1	1	1	1	1	1	1
	9		9	1	1	1	1	0	1	1
	blank		10	0	0	0	0	0	0	0


13.3.4. ソースコード

ソースコードはリスト 13-1 のとおりです。

リスト 13-1:List13-1_7Seg01.c

```
#include <stdio.h>          //入出力
#include <stdlib.h>         //一般ユーティリティ
#include <wiringPi.h>       //wiringPi
#define DP 19              //小数点
//7セグメントLED 表示器 a, b, d, d, e, f, g
const int a_g[7] = {17,27,20,21,12,13,16}; //セグメントと GPIO の定義 ①
const int disp0_9[11][7] = {{1,1,1,1,1,1,0}, //0 表示パターンの定義
                             {0,1,1,0,0,0,0}, //1
                             {1,1,0,1,1,0,1}, //2
                             {1,1,1,1,0,0,1}, //3
                             {0,1,1,0,0,1,1}, //4
                             {1,0,1,1,0,1,1}, //5
                             {1,0,1,1,1,1,1}, //6
                             {1,1,1,0,0,0,0}, //7
                             {1,1,1,1,1,1,1}, //8
                             {1,1,1,1,0,1,1}, //9
                             {0,0,0,0,0,0,0} //ブランク
                            };
int main (void){
    int i,k;
    wiringPiSetupGpio();          //BCM の GPIO 番号を使用
    for(i=0;i<7;i++){           //セグメント a から g の GPIO を出力に設定
        pinMode(a_g[i], OUTPUT);
    }
    pinMode(DP,OUTPUT);          //dp(小数点)の GPIO を出力に設定

    while(1){ ④
        for(i=0;i<11;i++){      //表示パターンをセット
            printf("%d\\n",i);   //表示パターンの行番号をターミナルに表示
            for(k=0;k<7;k++){    //表示パターンのセグメント a から g を出力
                digitalWrite(a_g[k],disp0_9[i][k]);
            }
            delay(1000);         //1秒待つ
        }
    }
    return EXIT_SUCCESS;
}
```

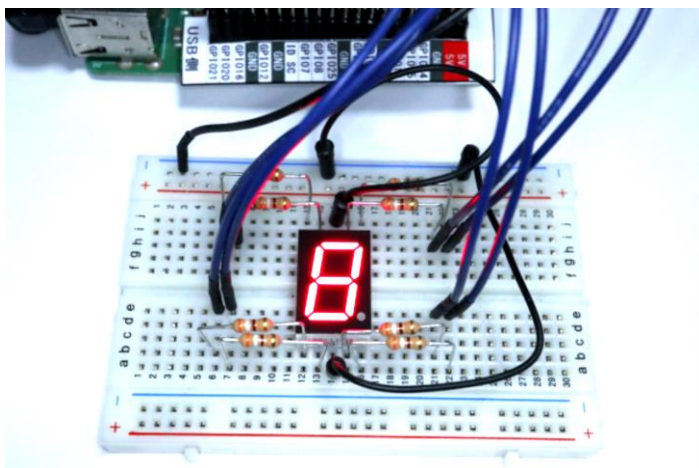


- ① a_g[7]配列で、セグメント a からgに配線する GPIO を定義します。
- ② disp0_9[11][7]2次元配列で、表 13-3 の表示パターンを定義します。
- ③ for 文を利用した二重ループになっています。変数 i は表 13-3 の表示パターンを指す行番号で、変数 k はセグメント指す列番号です。0 から 9 およびブランクを 1 秒毎に7セグメント LED 表示器に表示します。また、デバッグのため、表 13-3 の行番号をターミナルに表示しています。
- ④ while(1)は永久ループです。

13.3.5. 動作確認

リスト 13-1 をビルドして実行します。7 セグメント LED 表示器に、「0」から順番に 1 秒毎にカウントアップして数字が表示されます (図 13-12)。

図 13-12 数字「8」を表示した7セグメント LED 表示器



＜動作しないときの対処方法＞

- デジタル数字の表示がおかしいとき、回路の配線とソースコードの表示パターンの定義を確認してください。
- 動作しないとき、サンプルコードをダウンロードして実行してみましょう。正常に動作した場合は、ユーザーのソースコードに不具合がある可能性があります。
- サンプルコードも動作しない場合、配線した回路に問題があるかもしれません。配線間違いを確認しましょう。また、抵抗値も確認してください。
- 7セグメント LED 表示器のタイプがカソードコモンであることを確認します。本書では、7セグメント LED 表示器に C-551SRD (秋月電子通商:I-00640) を使用していますが、他の製品の場合にはピン配置が異なることがありますので、データシートを確認してください。
- Chapter11 のシャットダウン「List11-7_Shutdown」のプログラムを/etc/rc.local に登録している場合は、次のようにコメントアウトしてください。本回路に使用している GPIO17 が自走ロボットのシャットダウンボタンと兼用しているため、リスト 13-1 を実行するとラズパイがシャットダウンします。

```
#sudo /home/pi/MyApp/List11-7_Shutdown &
```

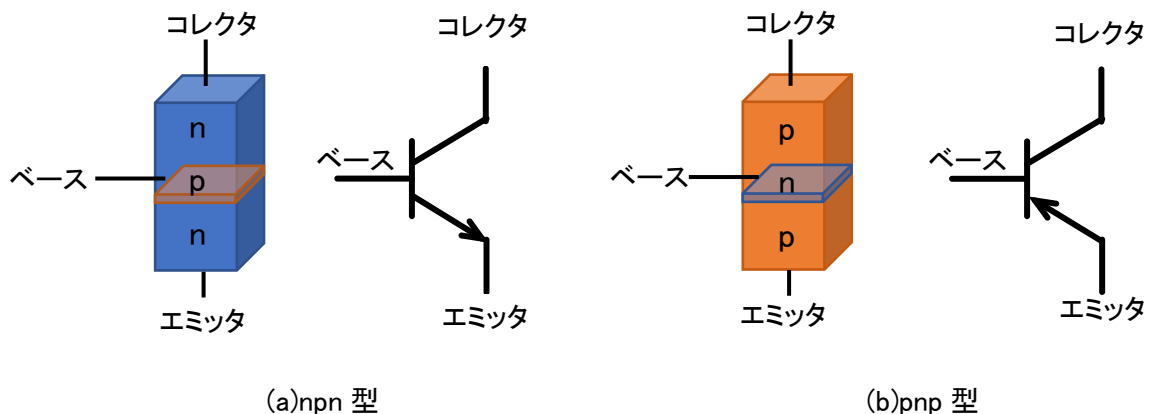
13.4. トランジスタとは

4 桁 7 セグメント LED 表示器をダイナミック点灯方式で表示する回路に、トランジスタ (Transistor) を使用します。本節では、トランジスタの概要を説明します。

トランジスタには小さな信号を大きな信号に変換する**増幅作用**と、信号を ON/OFF する**スイッチング作用**の機能があります。例えば、スピーカの音量の調整に増幅作用が使われ、H と L のデジタル信号にスイッチング作用が使われています。これから説明するダイナミック点灯方式の回路では、トランジスタのスイッチング作用を使用します。

トランジスタはダイオードと同じ、**p 型半導体**と**n 型半導体**で構成され、**npn 型**と**pnp 型**の 2 種類があります(図 13-13)。トランジスタには**ベース(Base)**、**コレクタ(Collector)**、**エミッタ(Emitter)**の 3 つ領域があり、それぞれリードに接続されています。エミッタの矢印は電流が流れる方向を示しています。コレクタとエミッタ間に流れる電流の効率を良くするために、ベース領域の幅は薄い構造になっています。

図 13-13 トランジスタの構造と電気用図記号

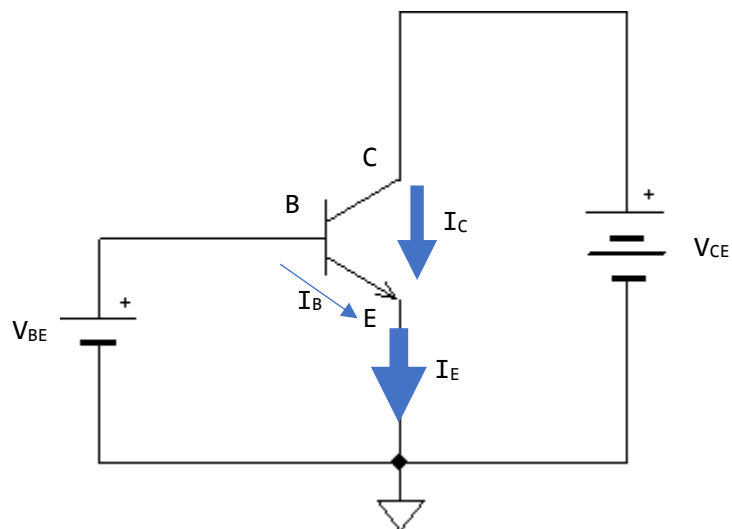


トランジスタの動作を npn 型を例に解説します(図 13-14)。トランジスタにベースエミッタ間電圧 V_{BE} とコレクタ・エミッタ間電圧 V_{CE} を与えると、**ベース電流 I_B** と**コレクタ電流 I_C** が流れます。コレクタ電流 I_C はベース電流 I_B に**直流電流増幅率 h_{FE}** と呼ばれる係数を乗じた値になります(式 13-1)。直流電流増幅率 h_{FE} はトランジスタによって異なりますが数十倍から数百倍です。よって、ベース電流 I_B の小さな変化から、大きなコレクタ電流 I_C を作り出すことができます。これがトランジスタの**増幅作用**です。**エミッタ電流 I_E** は、ベース電流 I_B とコレクタ電流 I_C を加算した値になります(式 13-2)。

$$I_C = h_{FE} \times I_B \quad [\text{式 13-1}]$$

$$I_E = I_B + I_C \quad [\text{式 13-2}]$$

図 13-14 電流の増幅作用



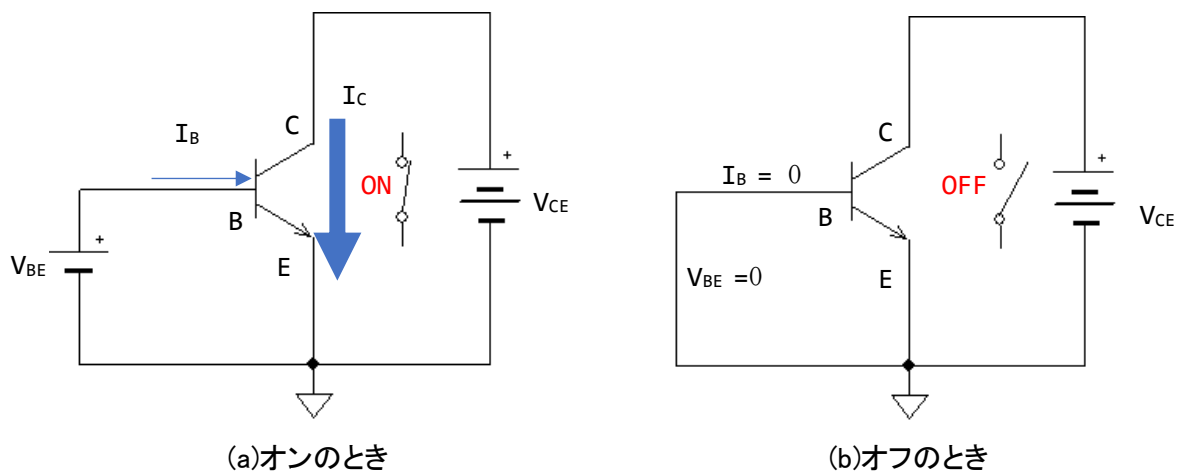
次に、**スイッチング作用**について説明します。スイッチング作用とは、トランジスタがスイッチと同じ働きをすることです。

- トランジスタをオンさせるには、図 13-15(a)のように、トランジスタにベースエミッタ間電圧 V_{BE} とコレクタ・エミッタ間電圧 V_{CE} を与えます。すると、ベース電流 I_B が流れて大きなコレクタ電流 I_C が流れます。これにより、コレクタとエミッタ間が導通して、スイッチが ON の状態と同じになります。

- 次に、トランジスタをオフさせるには、図 13-15(b)のように、トランジスタにコレクタ・エミッタ間電圧 V_{CE} のみ与え、ベースエミッタ間電圧 V_{BE} を 0V にします。 V_{BE} が 0V なので、ベース電流 I_B もゼロです。コレクタに電圧 V_{CE} が印加されていますが、コレクタ電流 I_C を流すことができないため、コレクタとエミッタ間は非導通になります。従って、スイッチが OFF の状態と同じになります。

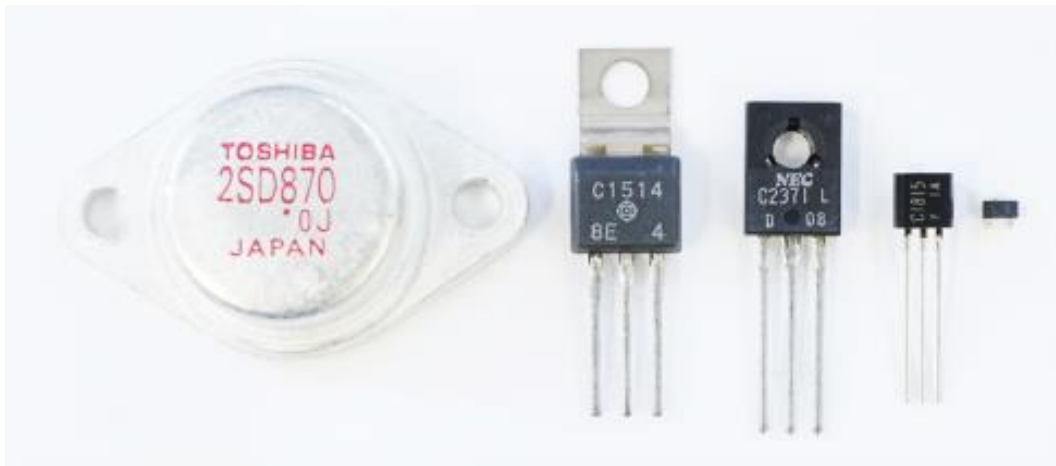
このように、ベース電流 I_B により、コレクタとエミッタ間をオンにしたり、オフしたりの制御ができるのです。

図 13-15 スwitchング作用のオンとオフ



トランジスタは用途に応じて、大小様々な種類が製品化されています（図 13-16）。

図 13-16 様々なパッケージのトランジスタ



トランジスタは 1947 年に米国ベル研究所で発明されました。1956 年にトランジスタの発明に貢献したジョン・バーディーン(John Bardeen)⁴、ウォルター・ブラッテン(Walter Brattain)、ウィリアム・ショックレー(William Shockley)の 3 名はノーベル物理学賞を受賞しました。トランジスタの発明により、半導体の研究が盛んになり、様々な研究者によって MOSFET(電界効果トランジスタの一種)などの新しいデバイスが開発されたり、製造技術が発展したりして、小さなシリコンチップにたくさんのトランジスタを集積して電子回路を製造することが可能になりました。それを集積回路(IC:integrated circuit)と呼びます⁵。1971 年に、嶋正利(しま まさとし)がインテル社に出向してインテルの社員と開発した世界初のマイクロプロセッサ i4004 は約 2,300 個のトランジスタを集積して作られました。今日のマイクロプロセッサは 10 億個以上のトランジスタを集積しています。

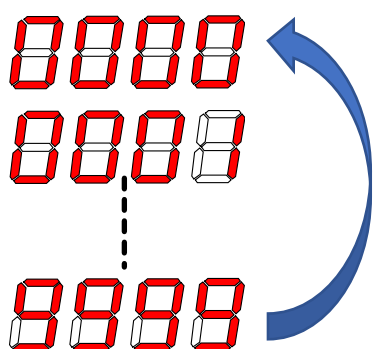
⁴ 1951 年、イリノイ大学の教授になり新しい研究に着手した。1972 年に超電導の研究で 2 度目のノーベル物理学賞を受賞した。

⁵ 1958 年に米国のジャック・キルビー(Jack Kilby(当時、TI(テキサス・インスツルメンツ)技術者))が IC を発明した。その功績により、2000 年にノーベル物理学賞を受賞した。

13.5. 4桁の数字をダイナミック点灯させる

4桁の7セグメントLED表示器を図13-17に示すように、「0000」から順番にカウントアップして表示して「9999」になったら、また、「0000」から繰り返し表示するプログラムを作成します。図13-7の1個の7セグメントLED表示器を表示させる回路より、セグメントaからgまでの7本のGPIO信号が必要になります。7セグメントLED表示器の桁数が増えると、ラズパイのGPIOが足らなくなってきます。桁数が多い場合、一般的にはダイナミック点灯方式の回路が採用されます。ダイナミック点灯方式には、どんな工夫があるのでしょうか。

図13-17 4桁の7セグメントLED表示器でカウントアップして繰り返す動作



13.5.1. 4桁7セグメントLED表示器の構造

図13-18に示すように4個の7セグメントLED表示器を内蔵した製品があります。本書では、カソードコモンタイプのOSL40562-LR（秋月電子通商：I-03673）を使用します。

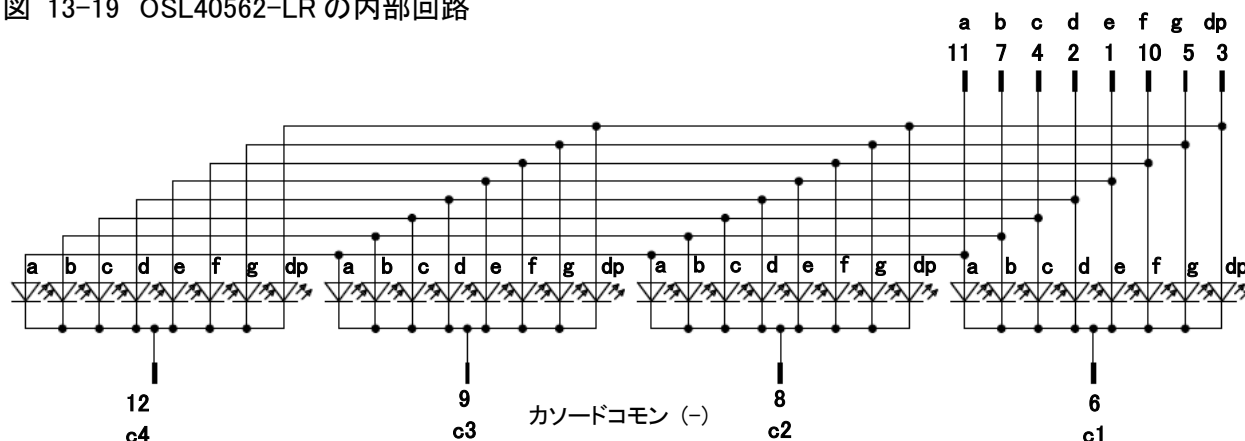
OSL40562-LRの内部回路図を図13-19に示します。7セグメントLED表示器ごとにカソード側で共通（コモン）に結線がされ、C1は1桁目、C2は2桁目、C3は3桁目、C4は4桁目のカソードコモン(-)になります⁶。アノード側はセグメント毎に共通に結線されています。そのため、セグメントに加えられた電圧が各桁に共通して通電されますので、4桁とも同じ数字しか表示されません。実は、OSL40562-LRはダイナミック点灯方式で回路設計されることを前提に製造されています。

図13-18 OSL40562-LRの表示面



⁶ OSL40562-LRのデータシートのコモンの名称を本書の解説に合わせるために変更しています。

図 13-19 OSL40562-LR の内部回路



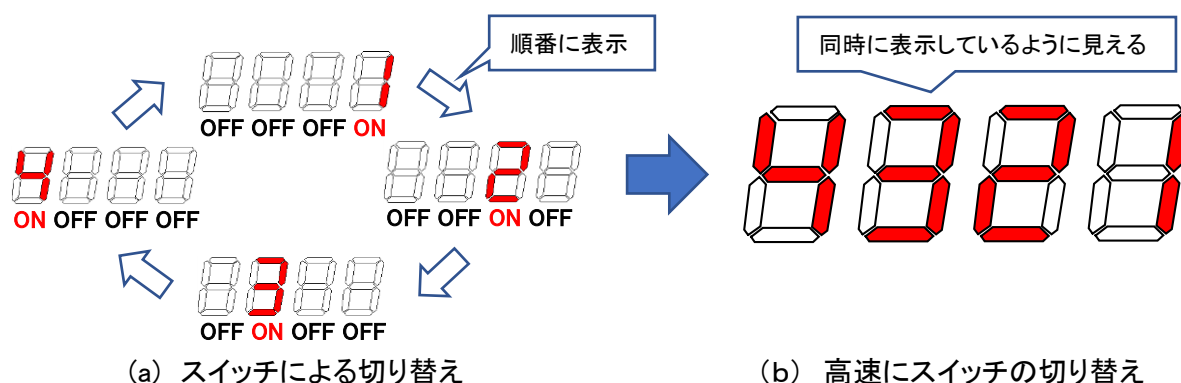
13.5.2. ダイナミック点灯方式とは

光を見ているとき、対象が消えたあとも、しばらくは、それを見続けているように錯覚する現象を**残像効果**⁷といいます。**ダイナミック点灯方式**は残像効果を利用して、複数の LED が同時に点灯しているように知覚させる方式です。ここでは、4 桁 7 セグメント LED 表示器を例に解説します（図 13-20）。

4 桁 7 セグメント LED 表示器のカソードコモンにそれぞれスイッチを付けたとします。表示させる桁のみスイッチを ON にして、残りの桁はスイッチ OFF にして非表示にします。ゆっくとスイッチを切り替えれば各桁が順番に表示して見えます（図 13-20(a)）。

ここでスイッチを高速に切り替えます。例えば、1 桁目だけを 1ms 点灯させます。次に、2 桁目だけを 1ms 点灯させ、そして 3 桁目だけを 1ms 点灯させ、最後の 4 桁目を 1ms 点灯させたら 1 桁目へ戻り、繰り返します。残像効果により、図 13-20(b)のように 4 桁が同時に表示しているように見えるのです。

図 13-20 ダイナミック点灯方式の原理



ダイナミックとは**動的**と訳され、「時間によって状態や状況が変化する」の意味です。対義

⁷ 人の目の時間分解能は約 50ms から 100ms 程度であり、この時間よりも短い光の点滅は、連続点灯しているように知覚される。※出典: 「残像効果」『ウィキペディア日本語版』。2020 年 7 月 10 日(金)15:32 UTC、
URL <https://ja.wikipedia.org>

語はスタティック（**静的**）になります。図 13-7 の回路は**スタティック点灯方式**になります。

ダイナミック点灯方式の特徴は次の通りです。

- 少ない GPIO 信号で制御することが可能になります。例えば、4 桁 7 セグメント LED 表示器をスタティック点灯方式で制御する場合、4 桁で 4 倍になり、28 本の GPIO 信号が必要になります（表 13-4）。ダイナミック点灯方式では、7 個の LED を点灯させる信号と 4 桁を切り替える信号を必要とし、GPIO 信号は 11 本で済みます。
- ダイナミック点灯方式は一定の周波数で点灯させるので、スタティック点灯方式と比較すると回路の消費電力を抑えることができ、LED の寿命も長くなります。
- 欠点としては、回路が複雑になることと、点灯の周波数が遅くなると表示にチラツキが発生します。

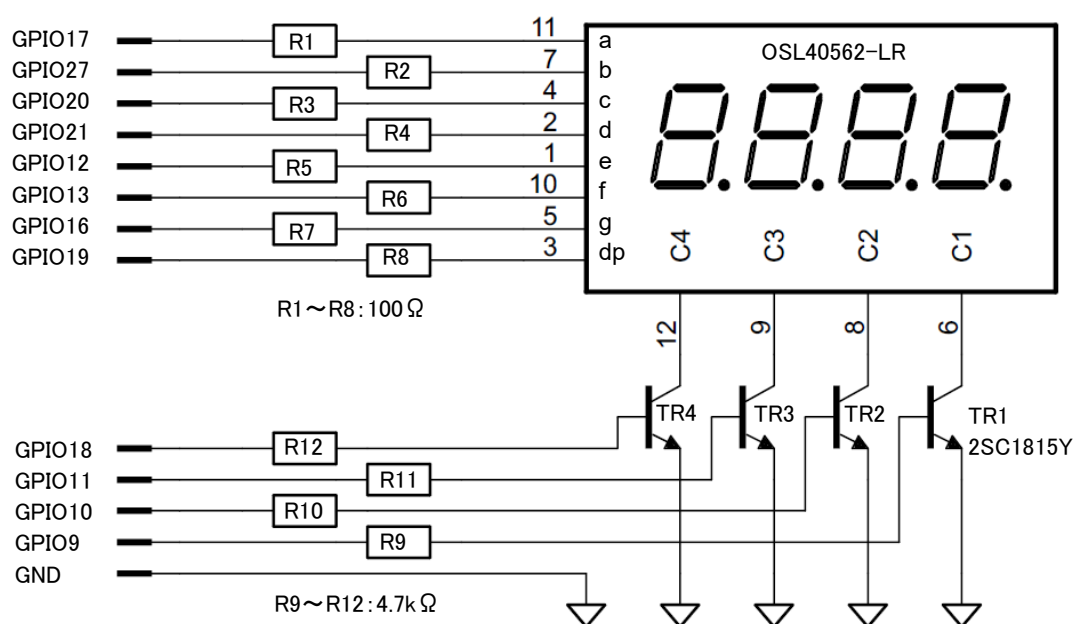
表 13-4 4 桁 7 セグメント LED 表示器に必要な信号数

スタティック点灯方式	ダイナミック点灯方式
7(セグメント数) × N(桁数)	7(セグメント数) + N(桁数)
GPIO 信号 7 × 4 = 28 本	GPIO 信号 7 + 4 = 11 本

13.5.3. 回路図

ダイナミック点灯方式による 7 セグメント LED 表示器の回路を図 13-21 に示します。

図 13-21 ダイナミック点灯方式による 4 桁 7 セグメント LED 表示器の回路図



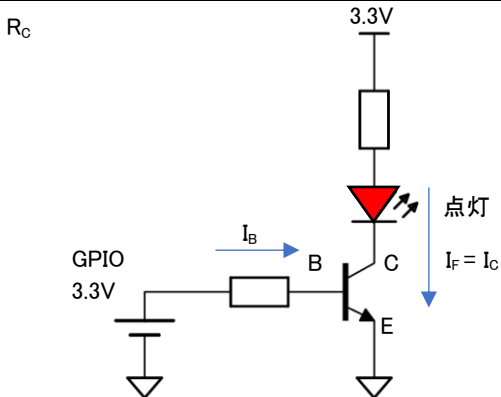
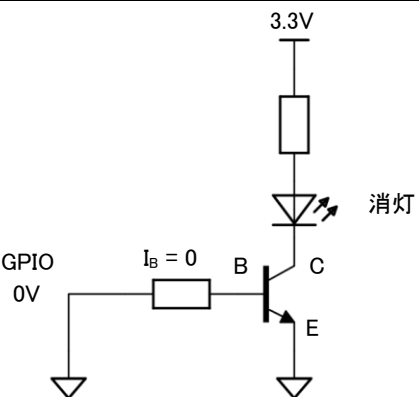
13.5.2 節では、図 13-20 で 4 桁 7 セグメント LED 表示器の各桁の切り替えに、スイッチを例

に説明しました。図 13-21 の回路では、そのスイッチをトランジスタで実現しています。トランジスタのスイッチング作用を使用します。

●トランジスタのスイッチング回路の動作

図 13-21 はトランジスタの**エミッタ接地回路**と呼ばれ、LED を点灯したり消灯したりする**スイッチング回路**として利用します。トランジスタのベースが入力となり、コレクタが出力になります。4 つのトランジスタ回路がありますが、同じ回路になります。1 つの回路の動作を表 13-5 に説明します。なお、図 13-21 では、GPIO の出力信号で抵抗 R_C を通して LED のアノード側に電圧を与えていますが、表 13-5 ではトランジスタの動作にフォーカスするため 3.3V に置き換えています。

表 13-5 トランジスタのスイッチング回路の動作原理

GPIO の出力信号が HIGH(3.3V)のとき	GPIO の出力信号が LOW(0V)のとき
トランジスタにベース電流 I_B が流れてトランジスタがオンの状態になります。コレクタとエミッタ間が導通状態になり、LED の順方向電流 I_F はコレクタ電流 I_C となり、グランド側へ流れて LED は点灯します。	トランジスタにベース電流 I_B は流れないため、トランジスタはオフの状態となります。コレクタとエミッタ間には非導通状態になり、LED の順方向電流 I_F は流れないため LED は消灯します。
	

● R_C と R_B の計算

図 13-21 に示した 4 桁 7 セグメント LED 表示器の回路から、1 桁を切り出した点灯回路を図 13-22 に示します。1 つのトランジスタで最大 8 個の LED を点灯します。

図 13-22 の抵抗 R_C と R_B の値を求めます。7 セグメント LED 表示器 OSL40562-LR のデータシートより、順方向電流 I_F を 7.5mA に設定します。ダイナミック点灯方式の場合、非表示の状態があるためスタティック点灯方式よりも少し電流を多くします。OSL40562-LR の順方向電圧 V_F の最大値は 2.3V です。また、トランジスタ 2SC1815 のデータシートより、コレクタ・エミッタ間飽和電圧 V_{CE} の最大値は 0.25V です。以上から、抵抗 R_C の電圧 V_{RC} が求まります。

$$V_{RC} = 3.3 - 2.3 - 0.25 = 0.75 \quad [\text{V}]$$

式 13-3 から、 R_C の抵抗値は 100 Ω と求まります。

$$R_C = \frac{V_{RC}}{I_F} = \frac{0.75}{7.5 \times 10^{-3}} = 100 \quad [\Omega] \quad \text{[式 13-3]}$$

ベース抵抗 R_B を求める手続きは、まずコレクタ電流 I_C からベース電流 I_B を求め、次に I_B からベース抵抗 R_B を求めます。

ベース電流 I_B はコレクタ電流 I_C と直流電流増幅率から式 13-1 より求めることができます。コレクタ電流 I_C は、8 個の LED を同時に点灯させた場合で求めます。よって、コレクタ電流 I_C は順方向電流 I_F 7.5mA の 8 倍の 60mA になります。

$$I_C = I_F \times 8 = 60 \quad [\text{mA}]$$

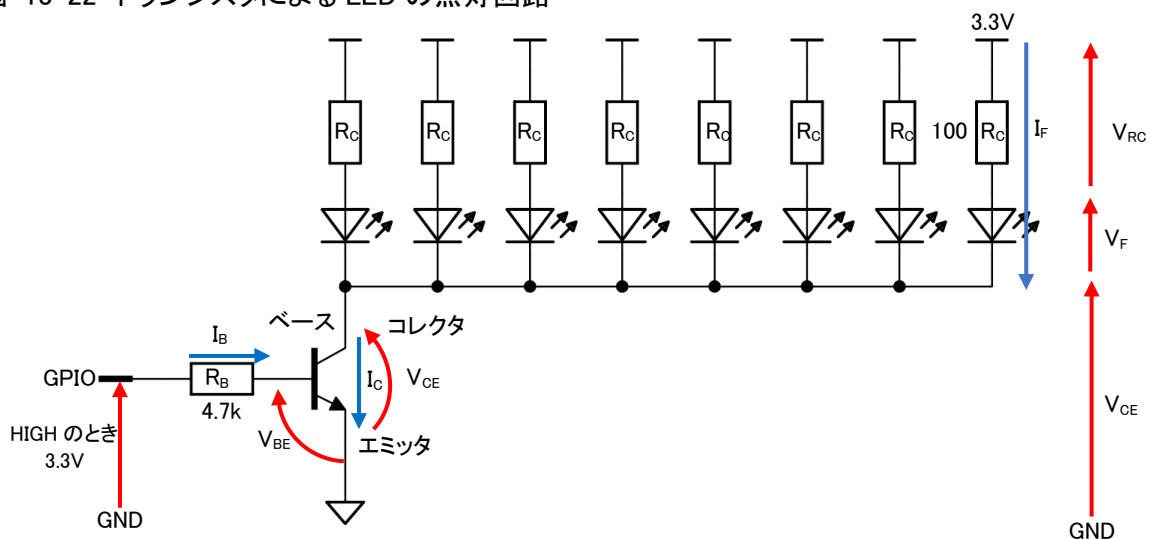
直流電流増幅率 h_{FE} は 2SC1815 のデータシートに明記されています。今回、トランジスタは 2SC1815 の Y クラスを使用します。Y クラスの直流電流増幅率 h_{FE} は 120 倍から 240 倍です。直流電流増幅率 h_{FE} は、最悪値となる最小値 (120 倍) を使用します。以上から、 I_B を式 13-1 より求めます。 I_B は 0.5mA となります。

$$I_B = \frac{I_C}{h_{FE}} = 0.5 \quad [\text{mA}]$$

次に、ベース抵抗 R_B を求めます。GPIO の出力信号 HIGH (3.3V) がベース端子に印加して、ベース電流 I_B が流れます。このベース電流はベースからエミッタへ流れます。このとき、ベース・エミッタ間電圧 V_{BE} が発生します。2SC1815 のデータシートより V_{BE} の最大値は 1.0V です。式 13-4 より R_B は 4.6k Ω と求められます。抵抗の E24 標準数値より、抵抗値を 4.7k Ω とします。

$$R_B = \frac{3.3 - V_{BE}}{I_B} = \frac{3.3 - 1.0}{0.5 \times 10^{-3}} \approx 4.6\text{k} \quad [\Omega] \quad \text{[式 13-4]}$$

図 13-22 トランジスタによる LED の点灯回路



13.5.4. 配線図

4桁7セグメントLED表示器 OSL40562-LR のピン配列（図 13-23）とトランジスタ 2SC1815 の端子名（図 13-24）を示します。配線図は図 13-25 になります。なお、本回路は新しいブレッドボードに組み立てます。

図 13-23 表示面から見た OSL40562-LR のピン配列

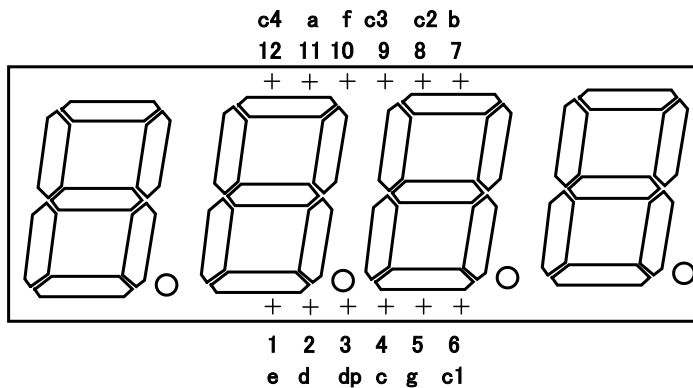


図 13-24 2SC1815 の端子名

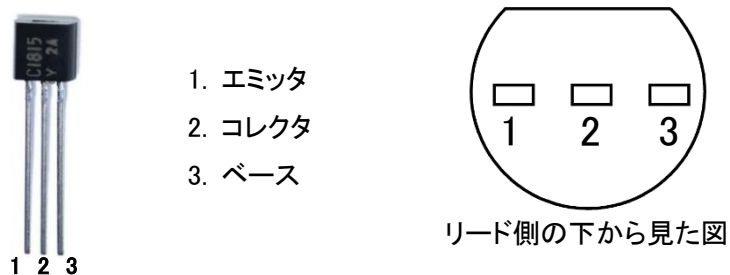
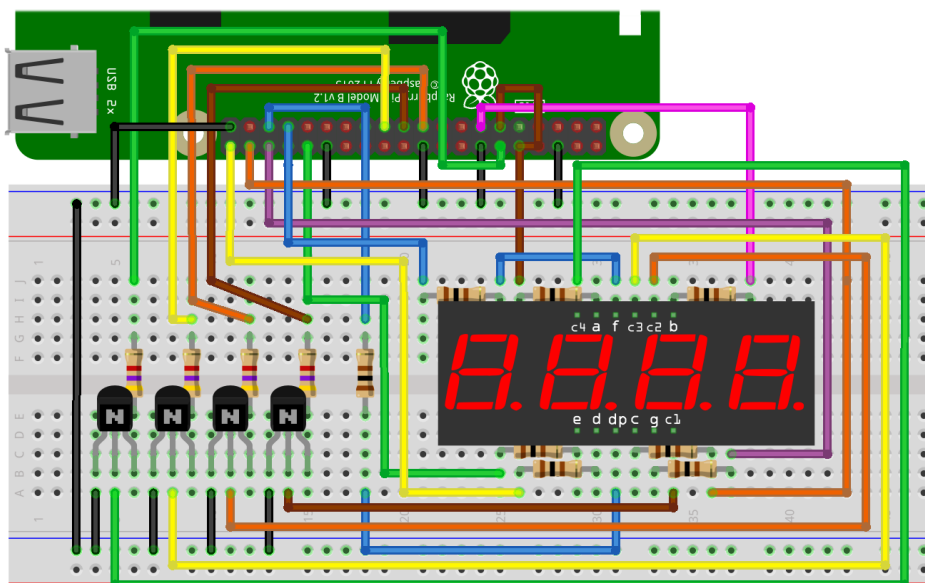


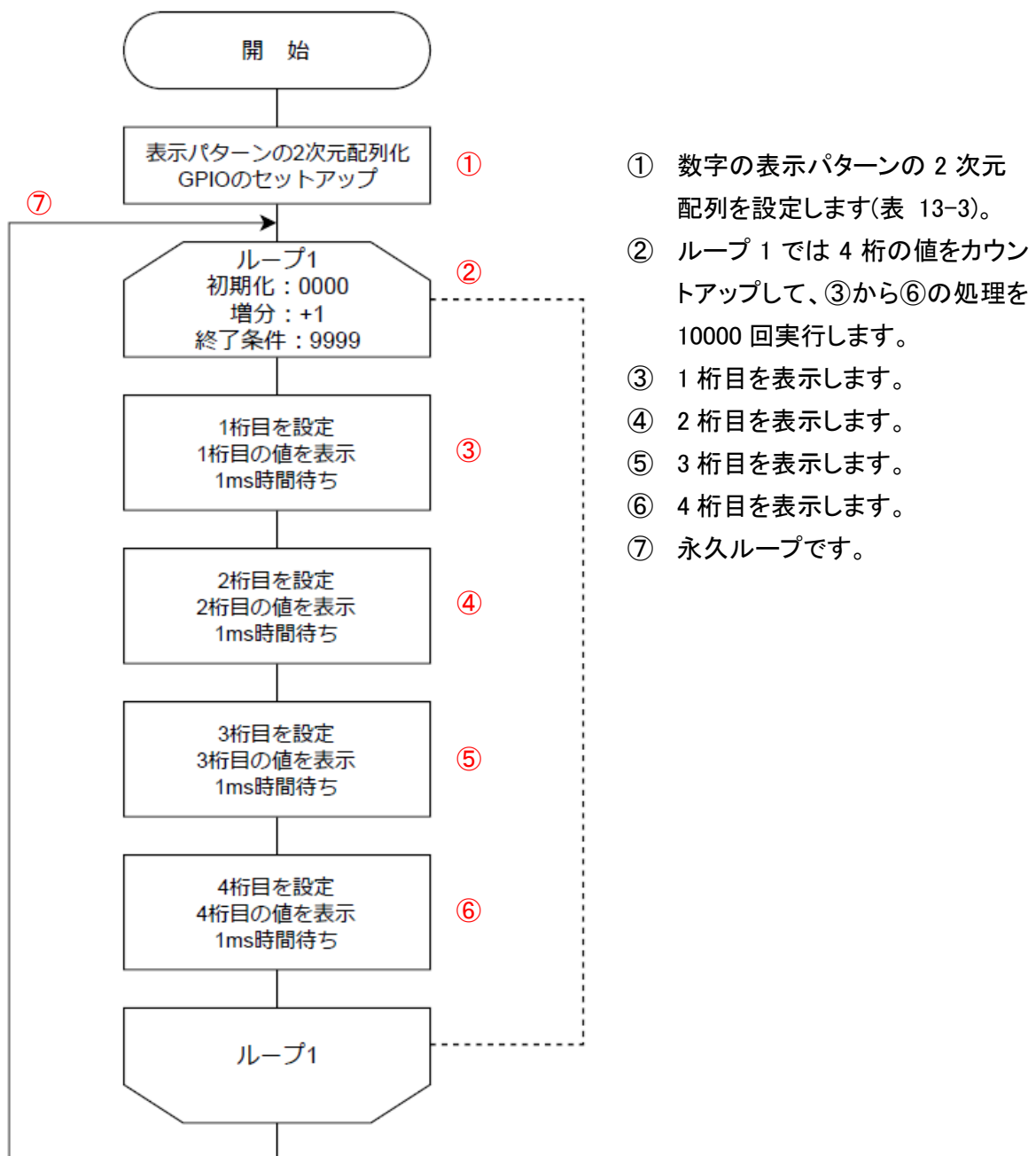
図 13-25 4桁7セグメントLED表示器の配線図



13.5.5. フローチャート

図 13-17 に示したように、4桁7セグメント LED 表示器に、「0000」からカウントアップして表示して、「9999」になったら、また、「0000」から繰り返し表示するプログラムを作成します。各桁を 1ms 毎に切り替えて表示するダイナミック点灯方式とします。4 桁を 4ms で表示したら、カウントアップします。フローチャートを図 13-26 に示します。

図 13-26 4 桁の数字を表示するフローチャート



13.5.6. ソースコード

ソースコードはリスト 13-2 のとおりです。

リスト 13-2: List13-2_7Seg02.c

```
#include <stdio.h>           //入出力
#include <stdlib.h>          //一般ユーティリティ
#include <wiringPi.h>        //wiringPi
#define DP 19                //小数点
#define INTERVAL 1          //各桁の点灯時間 ms
//com[0]1桁目,com[1]2桁目,com[2]3桁目,com[3]4桁目
const int com[4] = {9,10,11,18};
//7セグメントLED表示器 a, b, d, d, e, f, g
const int a2g[7] = {17,27,20,21,12,13,16}; //GPIOを配列で定義
const int disp0_9[11][7] = {{1,1,1,1,1,1,0}, //0
                             {0,1,1,0,0,0,0}, //1
                             {1,1,0,1,1,0,1}, //2
                             {1,1,1,1,0,0,1}, //3
                             {0,1,1,0,0,1,1}, //4
                             {1,0,1,1,0,1,1}, //5
                             {1,0,1,1,1,1,1}, //6
                             {1,1,1,0,0,0,0}, //7
                             {1,1,1,1,1,1,1}, //8
                             {1,1,1,1,0,1,1}, //9
                             {0,0,0,0,0,0,0}  //消灯
                            };

int main (void){
    int i,k;
    int segData;                //0000 から 9999 のカウント用
    int num1,num2,num3,num4;    //各桁の表示用
    wiringPiSetupGpio();        //BCMのGPIO番号を使用
    for(i=0;i<4;i++){           //4個のトランジスタを出力に設定
        pinMode(com[i],OUTPUT);
    }
    for(i=0;i<4;i++){           //4個のトランジスタを OFF
        digitalWrite(com[i],LOW);
    }
    for(i=0;i<7;i++){           //セグメント a から g の GPIO を出力に設定
        pinMode(a2g[i],OUTPUT);
    }
    pinMode(DP,OUTPUT);         //dp(小数点)のGPIOを出力に設定

    while(1){
        for(segData=0;segData<=9999;segData++){ ②
            //7Seg 1桁目
            digitalWrite(com[3],LOW); //4桁目 OFF
            digitalWrite(com[0],HIGH); //1桁目 ON ③
            num1=segData%10;           //10で割った余りから1桁目を取得 ④
            for(i=0;i<7;i++){         //セグメント a から g を出力
                digitalWrite(a2g[i],disp0_9[num1][i]);
            } ⑤
        }
    }
}
```

```

delay(INTERVAL);          //点灯時間 ⑥

//7Seg 2桁目
digitalWrite(com[0],LOW);  //1桁目 OFF
digitalWrite(com[1],HIGH); //2桁目 ON
k = segData/10;           //2桁目を1桁目へ
num2=k%10;                //10で割った余りから2桁目を取得
for(i=0;i<7;i++){        //セグメント a から g を出力
    digitalWrite(a2g[i],disp0_9[num2][i]);
}
delay(INTERVAL);          //点灯時間

//7Seg 3桁目
digitalWrite(com[1],LOW);  //2桁目 OFF
digitalWrite(com[2],HIGH); //3桁目 ON
k=segData/100;            //3桁目を1桁目へ
num3=k%10;                //10で割った余りから3桁目を取得
for(i=0;i<7;i++){        //セグメント a から g を出力
    digitalWrite(a2g[i],disp0_9[num3][i]);
}
delay(INTERVAL);          //点灯時間

//7Seg 4桁目
digitalWrite(com[2],LOW);  //3桁目 OFF
digitalWrite(com[3],HIGH); //4桁目 ON
k=segData/1000;           //4桁目を1桁目へ
num4=k%10;                //10で割った余りから4桁目を取得
for(i=0;i<7;i++){        //セグメント a から g を出力
    digitalWrite(a2g[i],disp0_9[num4][i]);
}
delay(INTERVAL);          //点灯時間

//ターミナルにデバック表示
printf("segData = %d %d %d %d %d\n",segData,num4,num3,num2,num1);
}
}
return EXIT_SUCCESS;
}

```

リスト 13-1 と重複する解説は割愛します。

- ① ダイナミック点灯用の4個のトランジスタを駆動するためにGPIOを出力に設定し、出力をOFFにして消灯させます。
- ② for文で、カウント値「0000」から「9999」までカウントアップします。
- ③ 表示させていた桁のトランジスタをOFFにし、次に表示させる桁のトランジスタをONします。
- ④ 4桁のカウント値から1桁目を求めます。例えば、カウント値が「4321」の場合、4321を10で割った余り1が1桁目に表示する値になります。C言語ではモジュロ演算子「%」で、余りを求めることができます。2桁目から4桁目も、桁を移動させて同様に求めることができます。表 13-6 に示すように、各桁を右へシフトするために10のべき乗で

割ります。その商を 10 で割った余りが表示する値になります。なお、4 桁目の「4 % 10」を計算すると商は「0.4」と思われますが、変数を整数型で計算すると商は「0」、余り「4」となります。

表 13-6 「1234」を各桁に表示させる手続き

桁	桁の右シフト	剰余算	余り(表示する値)
1 桁目	なし、または $4321 \div 10^0 = 4321$	$4321 \% 10$	1
2 桁目	$4321 \div 10^1 = 432$	$432 \% 10$	2
3 桁目	$4321 \div 10^2 = 43$	$43 \% 10$	3
4 桁目	$4321 \div 10^3 = 4$	$4 \% 10$	4

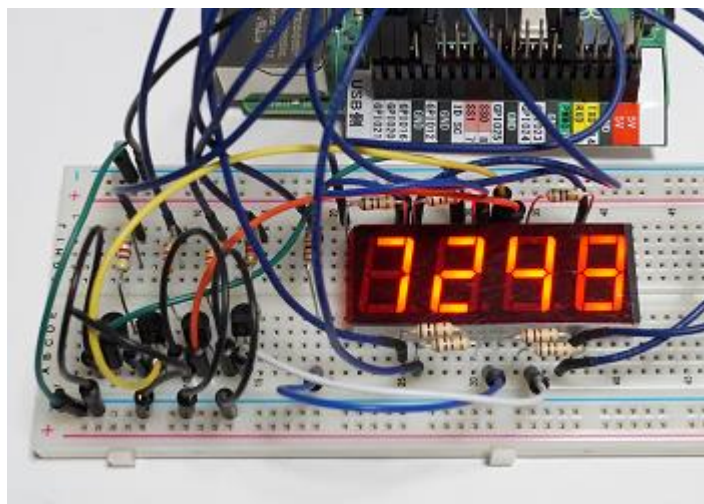
- ⑤ セグメント a から g を GPIO に出力します。
- ⑥ 1ms 間、該当桁の 7 セグメント LED 表示器を点灯させます。以降、カウント値の 2 桁目から 4 桁目を表示します。

13.5.7. 動作確認

リスト 13-2 をビルドして実行すると、ダイナミック点灯方式で4桁7セグメントLED表示器を「0000」から4ms毎にカウントアップして表示します。高速にカウントアップするため、1桁目は各セグメントが同時に点灯したように知覚して、数字の8に見えます(図13-27)。プログラムの実行を停止すると、1桁だけが点灯していたことが分かります。

また、リスト 13-2 の4桁毎のダイナミック点灯方式の周波数は約250Hzですが、ソースコード中のINTERVALの値を10にすると周波数は約25Hzとなり、7セグメントLED表示器の表示にチラツキが発生することを確認してください。なお、7セグメントLED表示器に単語帳などに付属する赤シートを貼ると数字が見やすくなります(図13-27)。

図 13-27 ダイナミック点灯している7セグメントLED表示器



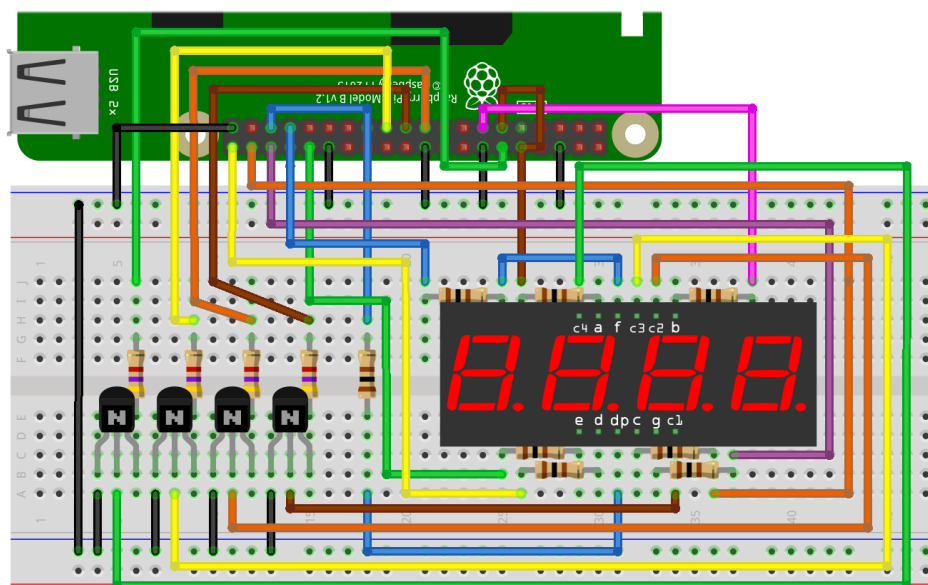
＜動作しないときの対処方法＞

- サンプルコードをダウンロードして、実行してみましょう。正常に動作した場合は、ユーザーのソースコードに不具合がある可能性があります。
- サンプルコードも動作しない場合、配線した回路に問題があるかもしれません。トランジスタや抵抗が増えて、配線が複雑になってきました。7セグメントLED表示器のピン番号、トランジスタの端子名、抵抗値、GPIOの番号等の配線を確認しましょう。
- 7セグメントLED表示器のタイプがカソードコモンであることを確認します。本書では、7セグメントLED表示器にOSL40562-LR(秋月電子通商:I-03673)を使用していますが、他の製品の場合にはピン配置が異なることがありますので、データシートを確認してください。

13.6. 章末問題

章末問題は、これまでの学習した内容を踏まえて応用力を習得するために用意しました。ここでは、4桁7セグメントLED表示器の配線図（図 13-25）を使用します。

図 13-28 章末問題の配線図



【問題 13-1】 List13-2_7Seg02.c を簡潔するプログラム

リスト 13-2 : List13-2_7Seg02.c のソースコードでは、4 桁を表示させるために似たような処理（図 13-26 フローチャートの③から⑥）を 4 回行っています。冗長な部分をひとまとめにしてみましょう。ファイル名は「List13_Ex01.c」とします。

- ヒント
 - List13-2_7Seg02.c の各桁の表示においてトランジスタを固定していますが、「徹底入門」p.96 の Led4bit 関数のように、トランジスタへの 4 本の GPIO 信号をひとまとめにしたポートとして扱うと、トランジスタの指定を汎化できます。
 - べき乗の計算には pow 関数があります。pow 関数を使用する場合は、ソースコードに数学ライブラリ<math.h>をインクルードし、ビルドオプションに数学関数オプションを意味する -lm (エルエム)を付けてください。

【問題 13-2】 16 進数でカウントアップするプログラム

List13-2_7Seg02.c または、List13_Ex01.c のソースコードを改造して、「0000」から 16 進数でカウントアップして表示して「FFFF」になったら、また、「0000」から繰り返し表示するプログラムを作成します。16 進数の A から F の表示パターンは図 13-3 を参照してください。ファイル名は「List13_Ex02.c」とします。

【問題 13-3】現在の時刻を表示するプログラム

現在の時刻を表示させるプログラムを作成します。4 桁目と 3 桁目に時を、2 桁目と 1 桁目に分を表示させます。ファイル名は「List13_Ex03.c」とします。

- ヒント

- 日付及び時間を取得するために time 関数、localtime 関数、tm 構造体を使用します。これらの関数や構造体は、ヘッダファイル time.h で定義されているので、インクルードします。

```
#include <time.h>
```

- time 関数でシステム時刻を取得しますが、変数は標準 C ライブラリが定義しているデータ型 time_t で宣言します。tm 構造体には、時(tm_hour)、分(tm_min)、秒(tm_sec)などの int 型メンバがあります。ここでは local で構造体のポインタ宣言をします。時と分を代入するために、hour と minute を用意します。

```
time_t now;
```

```
struct tm *local;
```

```
int hour, minute, second;
```

- time 関数の戻り値からシステム時刻を取得しますが、UNIX 時間とも呼ばれ協定世界時(UTC)での 1970 年 1 月 1 日午前 0 時 0 分 0 秒からの経過秒数です。localtime 関数は経過秒数からローカル時間(日本時間)に変換し構造体に格納します。アロー演算子(->)を利用して構造体から tm_hour(時)、tm_min(分)、tm_sec(秒)の構造体メンバを取り出します。

```
now = time(NULL);
```

```
local = localtime(&now);
```

```
hour = local -> tm_hour;
```

```
minute = local -> tm_min;
```

```
second = local -> tm_sec;
```

参考文献/参考資料

- [1] “C-551SRD-NW B/W データシート,” PARA LIGHT ELECTRONICS CO., LTD., (Aug. 2015)
- [2] Ian M. Ross, “The Invention of the Trasnsistor,” Proceedings of the IEEE, vol.86, no.1, pp.7-28, (Jan. 1998)
- [3] 嶋正利, “マイクロコンピュータの誕生—わが青春の 4004,” 岩波書店, (Aug. 1987)
- [4] “Four Digit Display OSL40562-LR データシート,” OptoSupply
- [5] “東芝トランジスタ 2SC1815 データシート,” TOSHIBA,(Jan.2002)
- [6] 押本愛之助,石坂陽之助,“トランジスタ回路演習(増訂版),” 工学図書, pp.209-218, (Jul. 1983)
- [7] 外山滋比古,“思考の整理学,” ちくま文庫, pp.60-62, (Apr. 1984)

シーべんご ラズベリーパイ てっぺいにゅうもん
C言語ではじめるRaspberry Pi 徹底入門
Chapter13 7セグメント LED 表示器で数字を表示する

2020 年 8 月 23 日 初版

2020 年 12 月 14 日 差し替え 図 13-27

2023 年 4 月 1 日 しおり付き PDF に変更

©2020 菊池達也