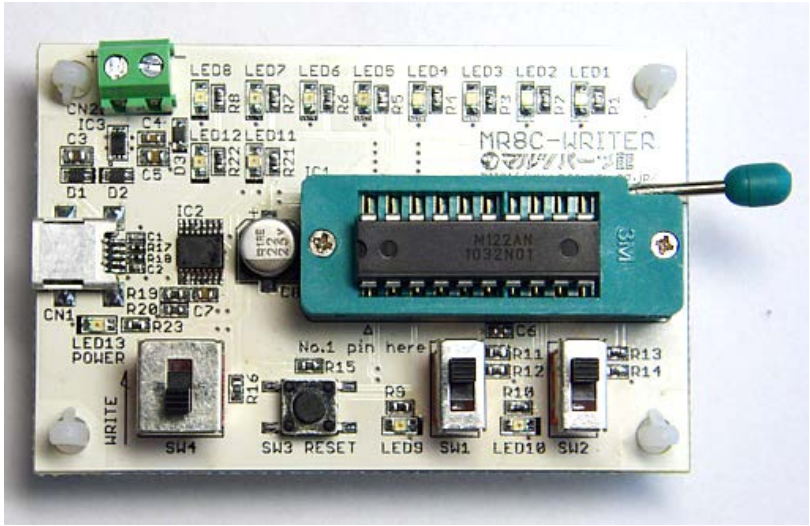
マイコン電子工作入門教室

マルツメイク館2013



はじめてのマイコン電子工作
「ライタ＆LEDピカピカ」

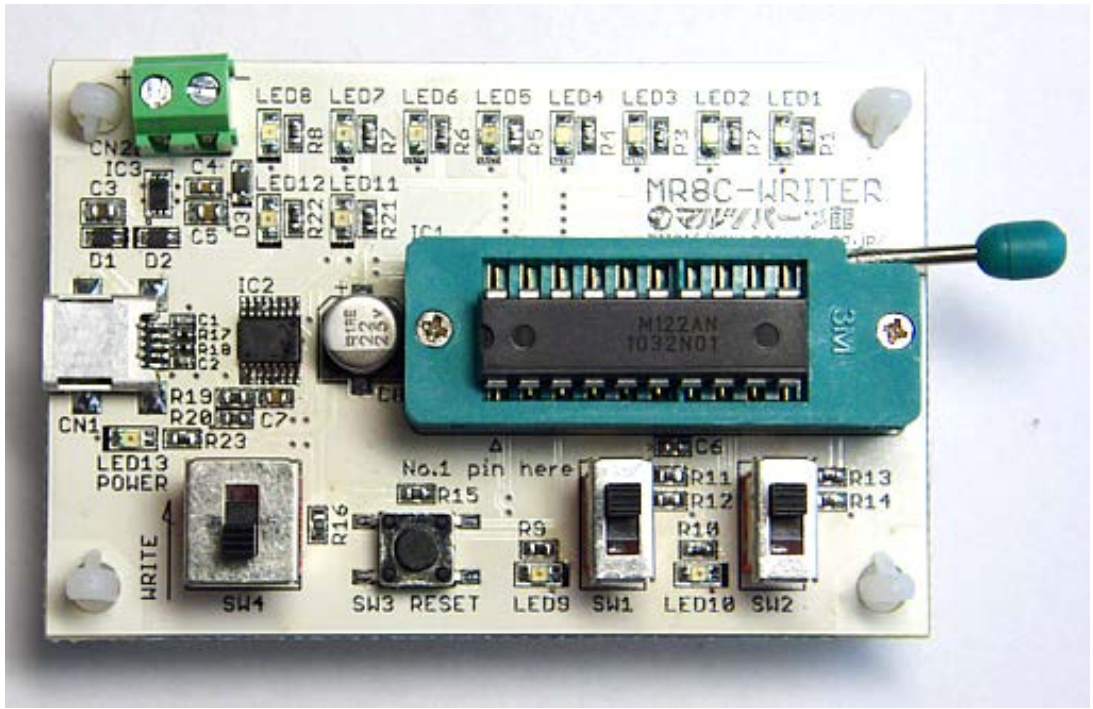
株式会社ルネサスソリューションズ

2013/ 7/06　Rev. 1.00

マルツパーツ館

2013　7月6日　メイク館電子工作講座

マイコン電子工作をたのしもう！



RENESAS

RENESAS R8C

RENESAS

電子工作教室　　(2013 7/06 13:00-17:00)

R8C/M12A ライタ＆LEDピカピカ キット

開発環境を整え、LEDを自由につけよう！

本日の予定

（1）オリエンテーション　配布品の確認　　60分
マイコンの原理と構成
LEDの点灯、定電流駆動とPWM制御

（2）M12Aライタ＆LEDピカピカ基板組み立てと動作確認　　60分
基板回路解説

（3）マイコン開発ツール　開発環境ソフト（HEWとコンパイラ）、書き込みソフトなど
必要ソフトのインストール
LED　Test　ソフトを書き込んでみよう！　　60分

（4）LEDピカピカ　プログラム（C言語）の解説！プログラムの変更　　60分
割り込みを使って、もっと正確に動作させる！
蛍の光。

1. オリエンテーション

・配布品の確認
・マイコンの原理と構成
・LEDの点灯、定電流駆動とPWM制御

ルネサスはマイコン世界一の会社です。

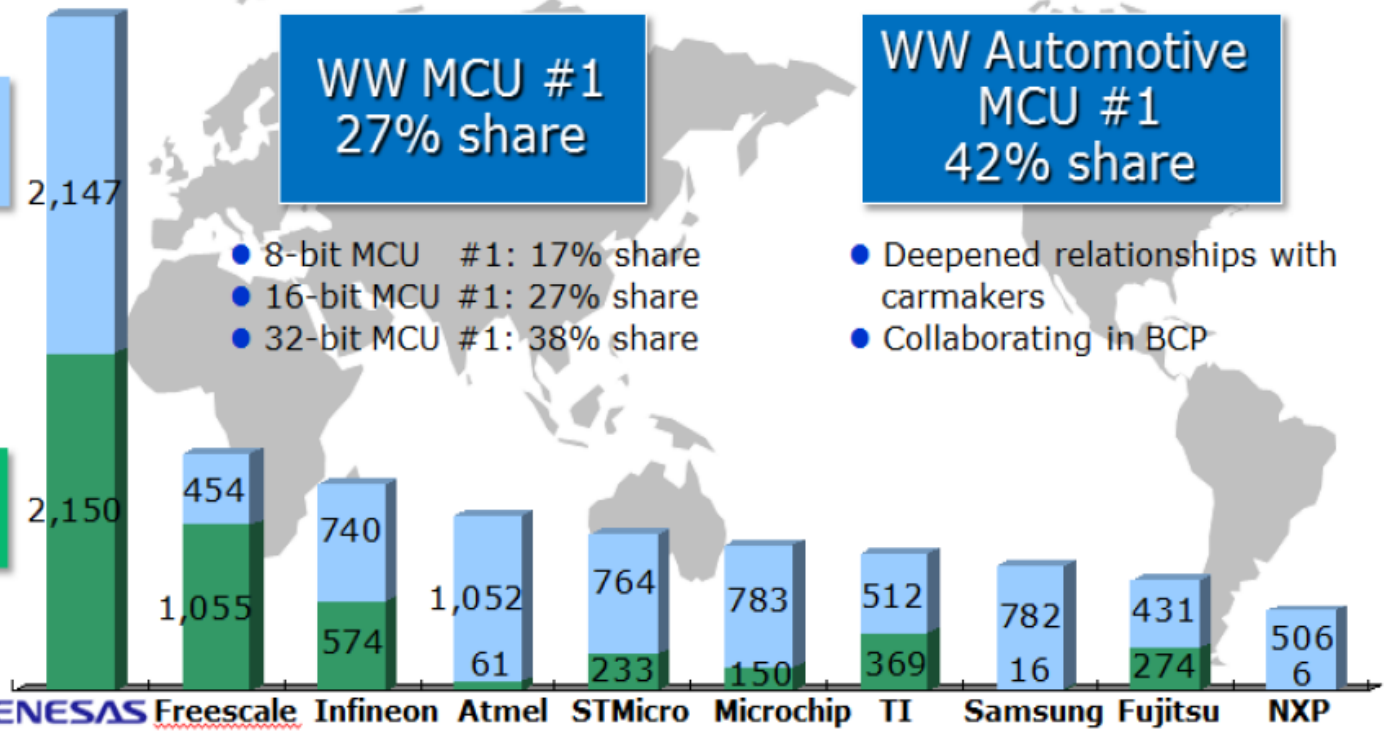
マイコンW/W売り

- Firmly maintaine 27% in CY2011
- Dominant positio world's No.1 sha

M US$
CY2011

General-purpose* MCU

Auto-motive MCU



WW MCU #1
27% share

- 8-bit MCU #1: 17% share
- 16-bit MCU #1: 27% share
- 32-bit MCU #1: 38% share

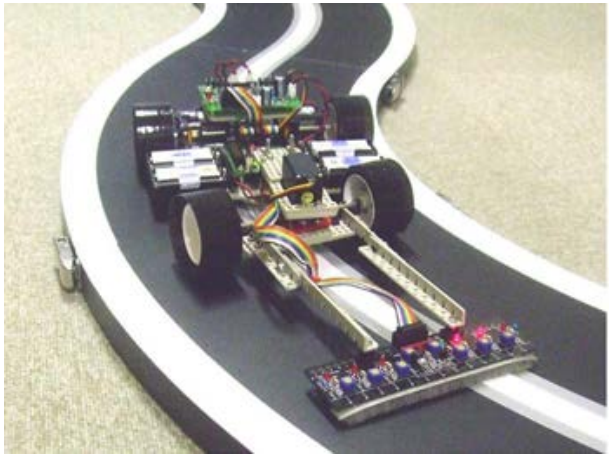
WW Automotive MCU #1
42% share

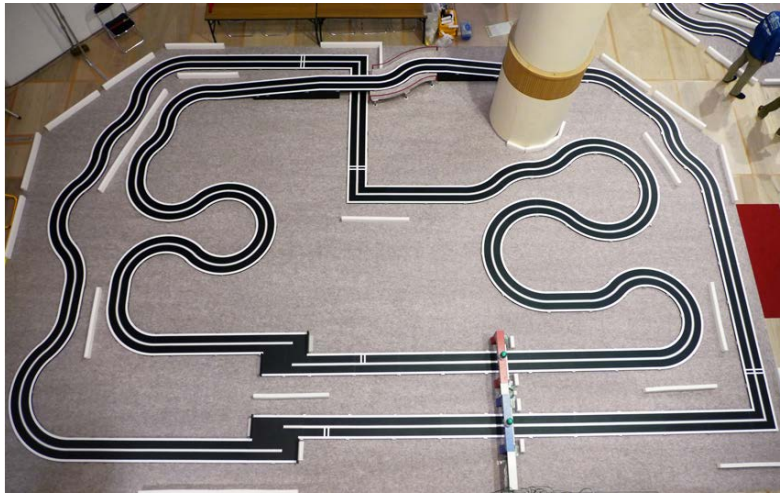
- Deepened relationships with carmakers
- Collaborating in BCP

*General-purpose MCU: MCUs for applications excluding automobiles

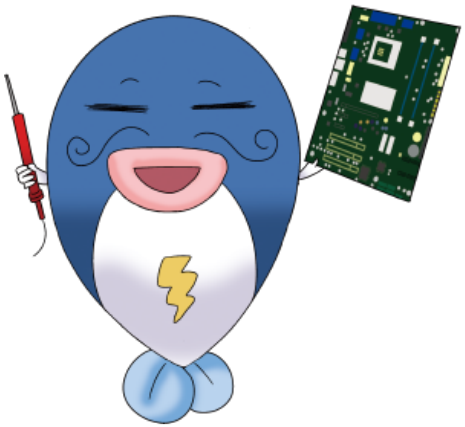
Source: IHS iSuppli, Annual 2011 Semiconductor Market Share

ジャパンマイコンカーラリー





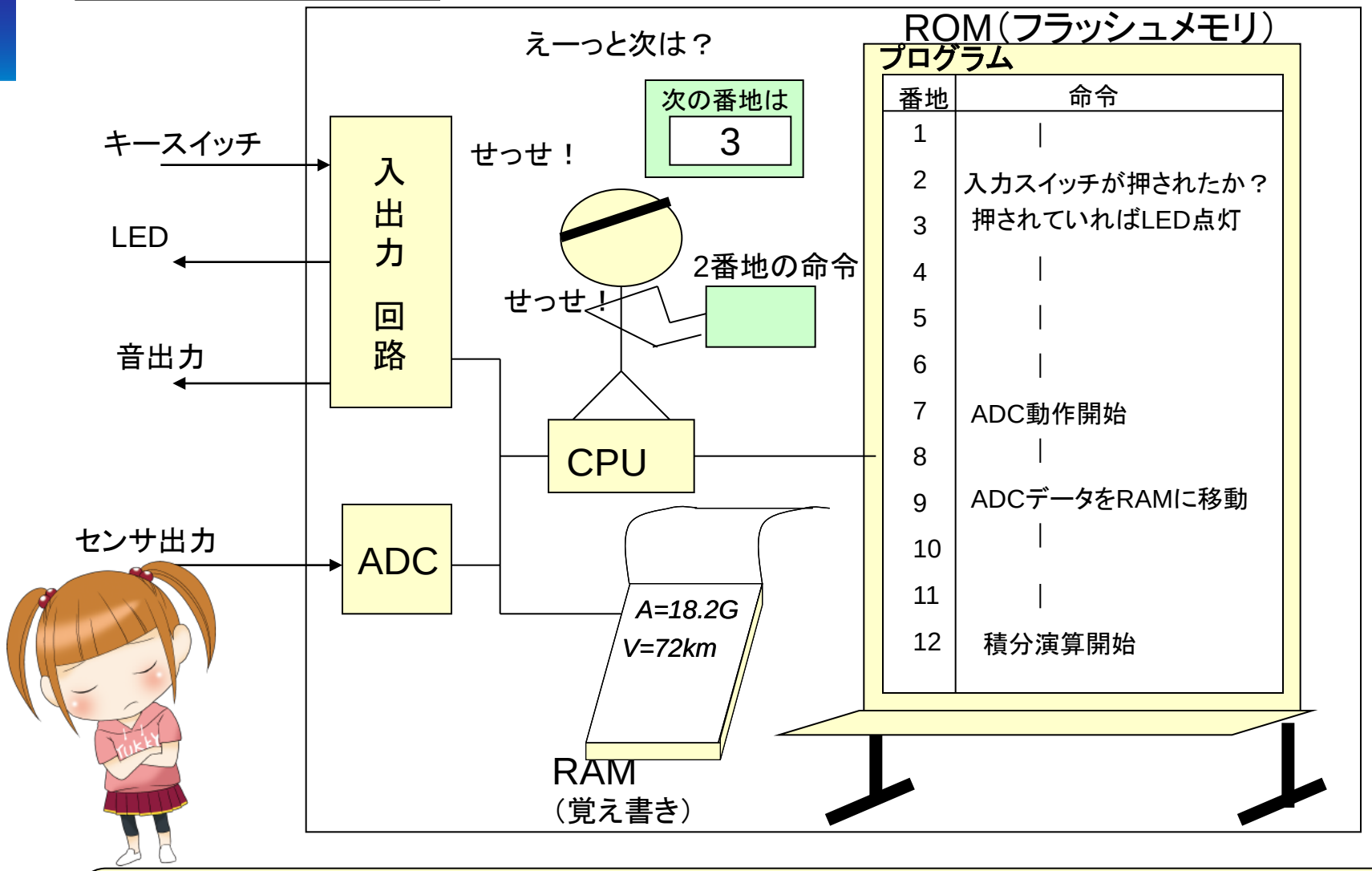
マイコン（マイクロコンピュータ）とは？



CPU（半導体の頭脳）、ROM（プログラムを入れておくメモリ）、RAM（計算用メモリ）、入出力制御回路、クロックパルス発生回路などが入っている半導体で

プログラムされたとおりに動作します。

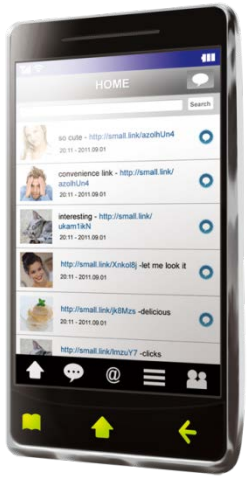
マイコンとは？



CPUは一番偉そうだけど、次の番地はと書いてあるプログラムカウンタの内容の番地の命令を取り出し(フェッチ)、内容を理解し(デコード)、忠実に実行しているだけです。
CPUに命令を出しているのは私たちが書いたプログラムなのです。命令の種類には入出力IFとデータをやり取りするものや、足し算や掛け算を行ってRAMとデータをやり取りするものなどがあります。

マイコンはどこに使われている?

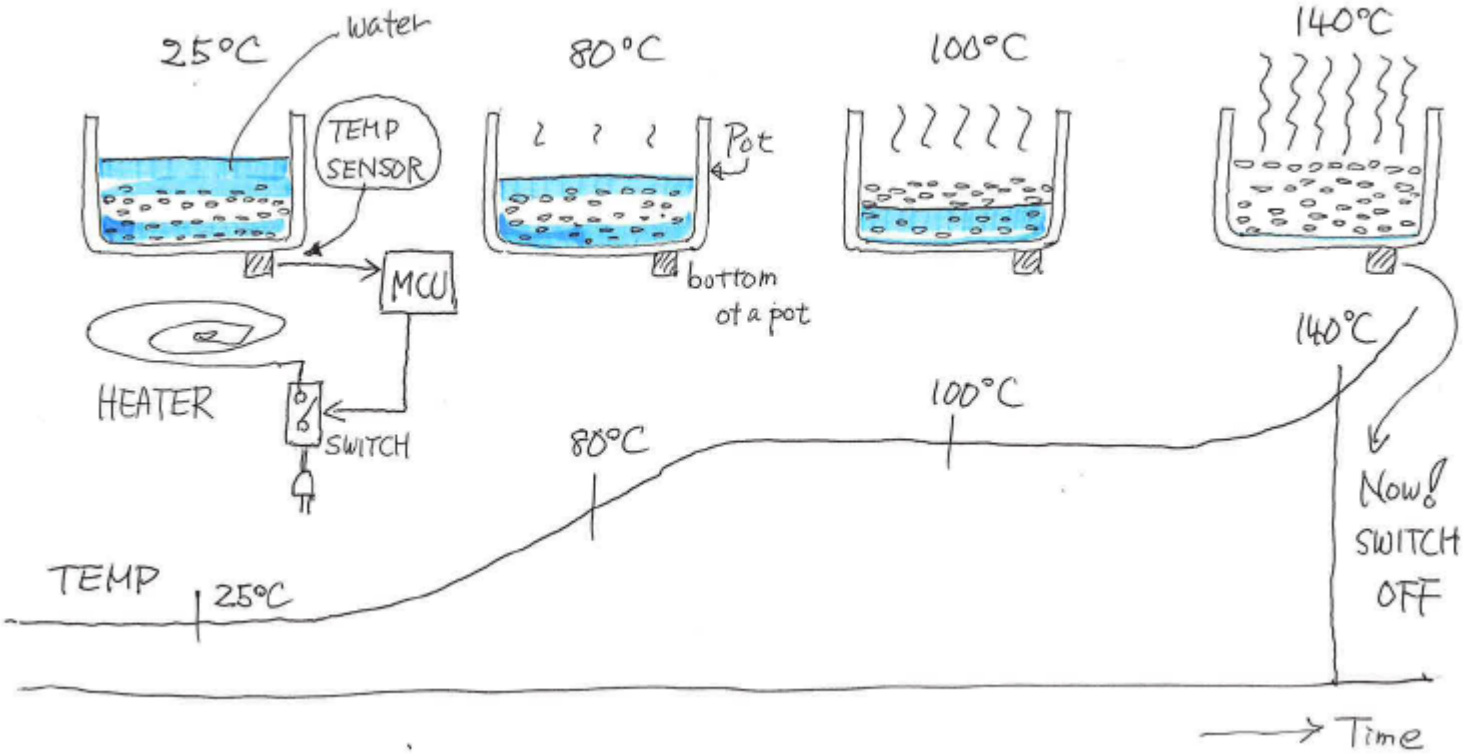
この中に入っているよ!



マイコンがご飯を炊ける原理

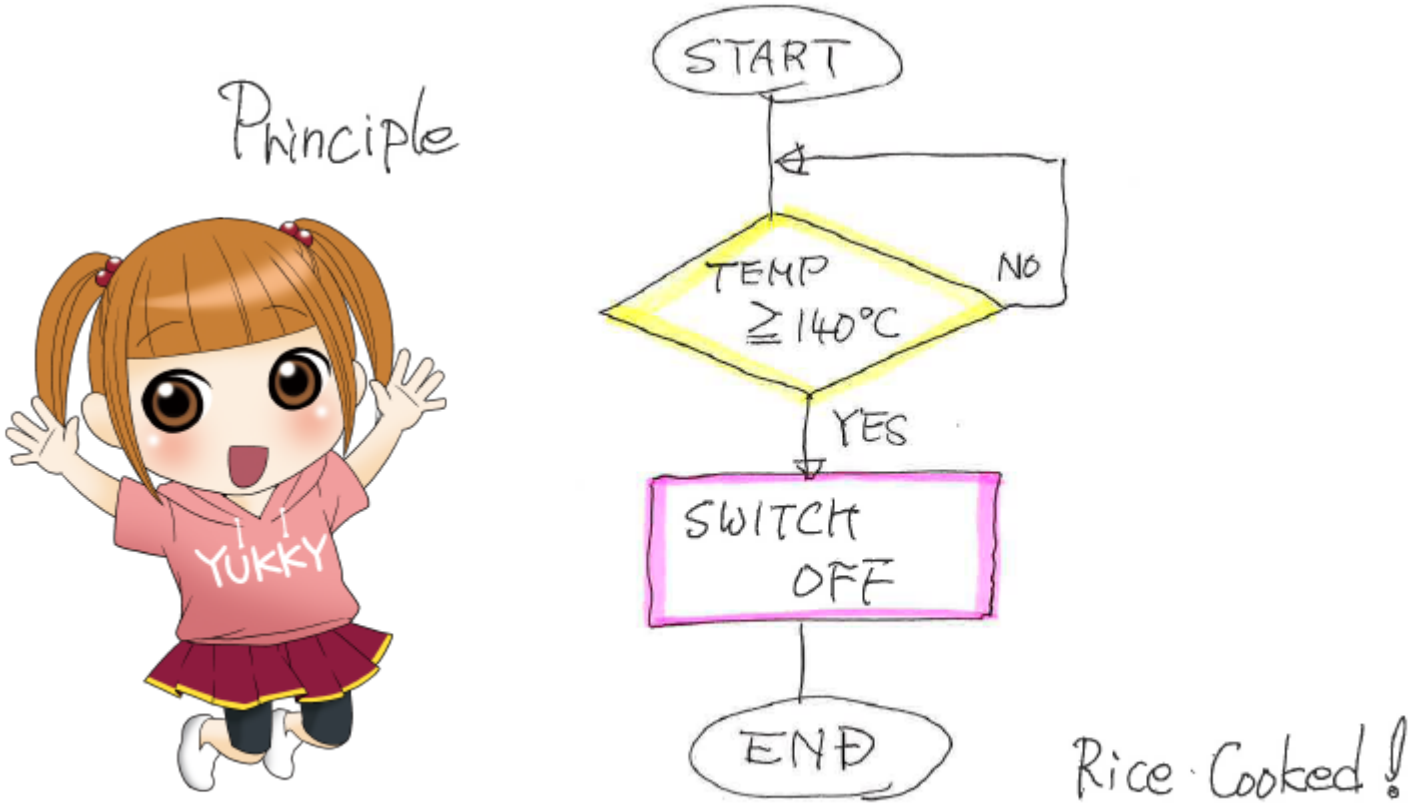
なぜお米の量にかかわらずご飯を炊けるのか?

水がある間は100度



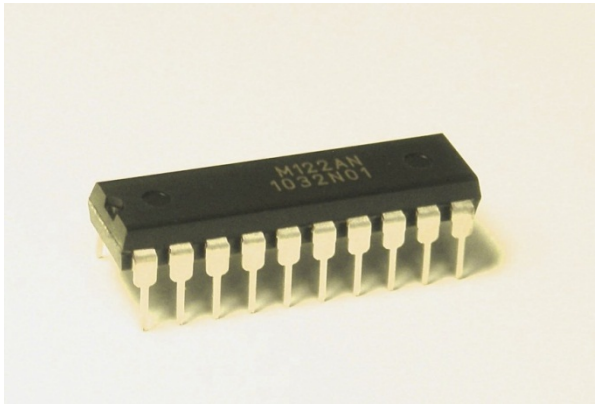
お釜の底の温度が140度になったら、マイコンはスイッチをオフにする

以上のようにフラッシュメモリにプログラムを書き込めば、マイコンに好きなことをさせられます。
例えば
下のようなプログラムを書けば炊飯器ができます。おもしろいですね。



ルネサスマイコンR8C/M12Aの特徴

電子工作愛好者のための20ピンDIP版マイコン



R8C/M12Aマイコン（R5F2M122ANDD）の特徴

- H8と同じ開発環境（HEW）が使える
- 16ビットCPU
- 20MHz高速オンチップオシレータ内蔵
- 16ビットタイマ　3本
- UART　1
- 10ビットA/Dコンバータ　6ch
- コンパレータ　2個
- 電源電圧　1.8-5.5V
- フラッシュROM　8kB、データフラッシュ　2kB、RAM　512B
- そして　安い！

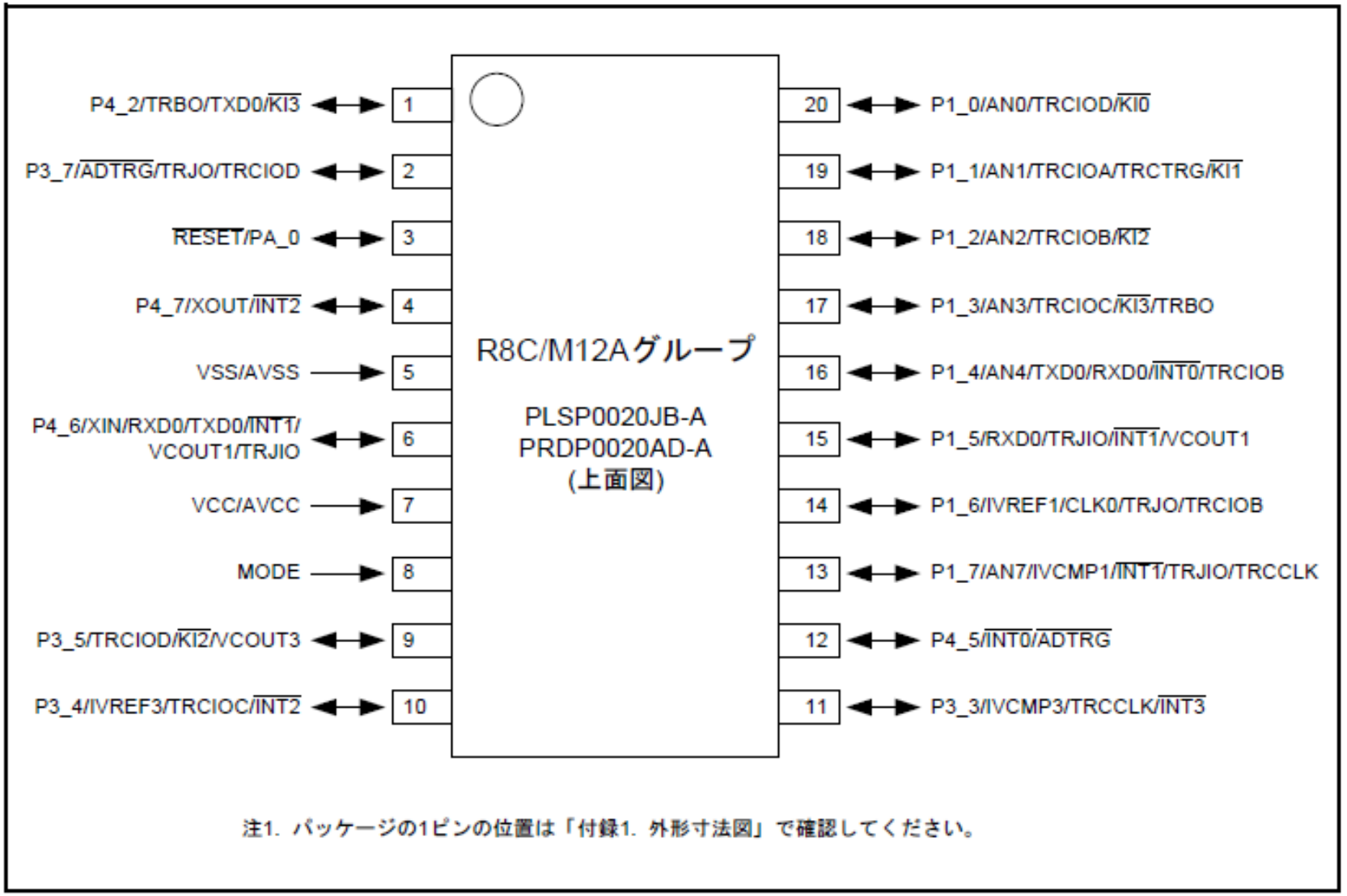


図1.4　R8C/M12Aグループのピン配置図(上面図)

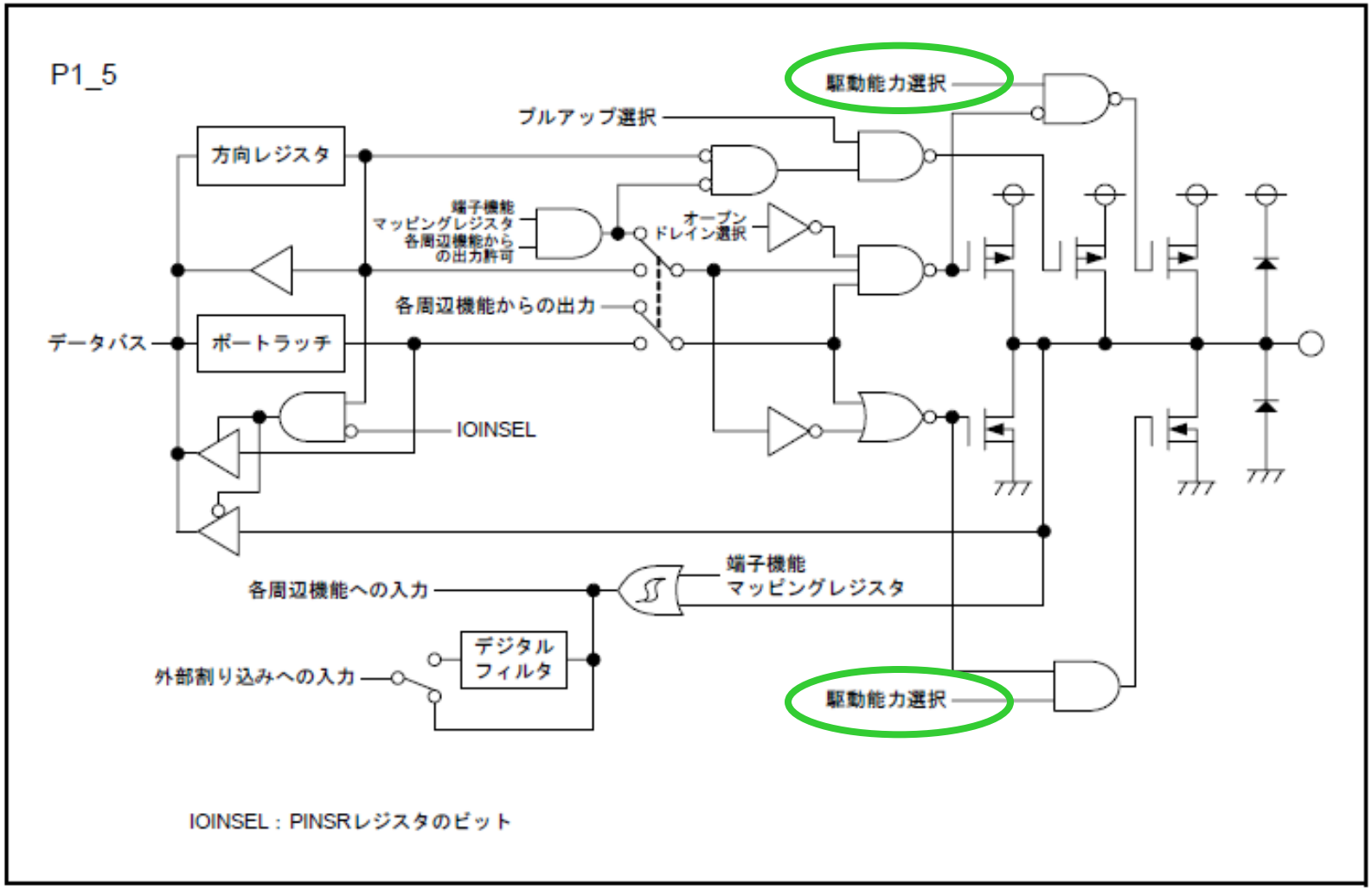
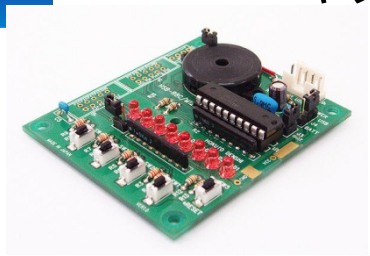
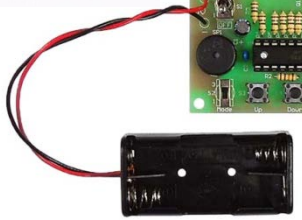
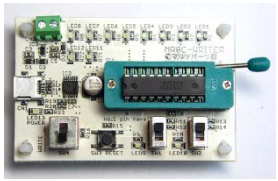


図12.8　I/Oポートの構成(3)

マイコン電子工作キット







R8C/M12A

マイコンレーサー

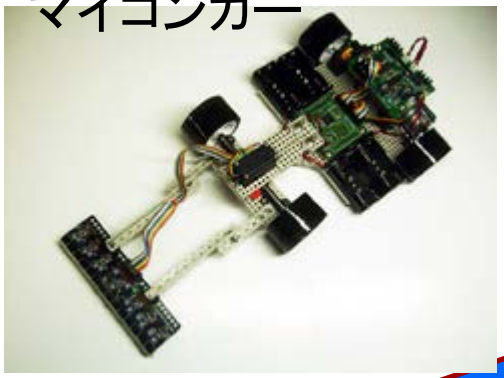
R8C/34C

ミニマイコンカー

R8C/35C

ものづくりコンテスト

マイコンカー



R8C/38C

R8Cマイコンシリーズ

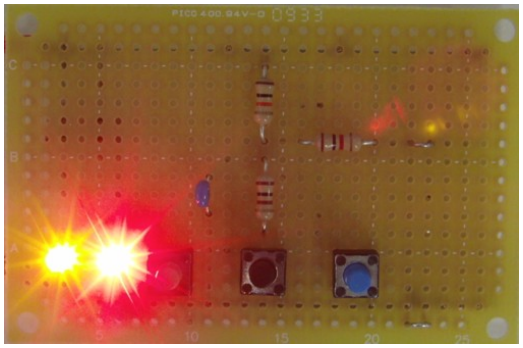
LED (Light Emitting Diode)

LEDの点け方　定電流駆動

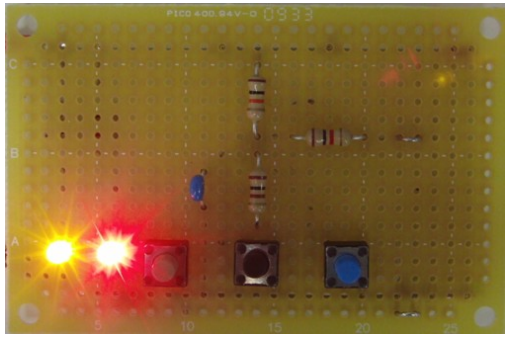
定格と電流制限、何mAでつくのか？　何mAで消える？
マイコンの端子で直接つけられる？
マイコンのポートの電気的特性

LEDの電流と明るさ

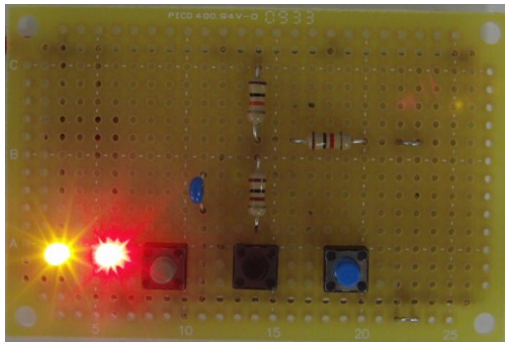
OSYL3133A（黄）、OSDK3133A（赤）、 Ifmax=30mA



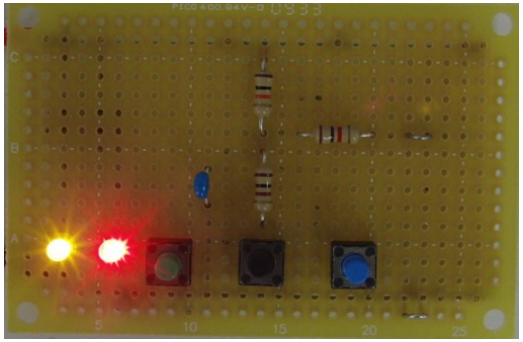
20mA



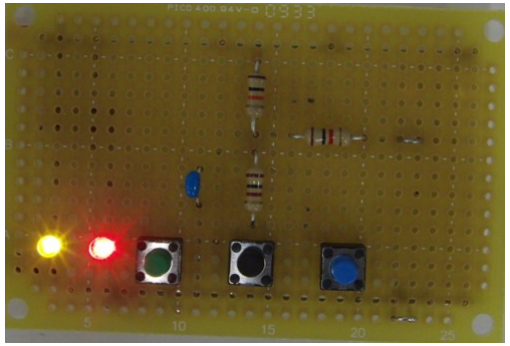
10mA



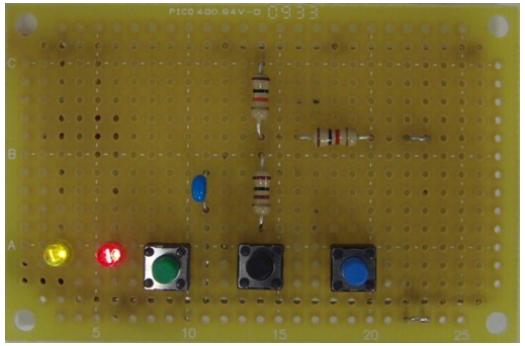
5mA



2mA



1mA



100uA

まだ点いている

ルネサスR8Cマイコンは LEDを直接駆動できる

平均5mAOK

表20.2　推奨動作条件

記号	項目		測定条件	規格値			単位
				最小	標準	最大	
Vcc/AVcc	電源電圧			1.8	—	5.5	V
Vss/AVss	電源電圧			—	0	—	V
VIH	H入力電圧	CMOS入力以外		0.8 Vcc	—	Vcc	V
		CMOS入力	4.0 V≦Vcc≦5.5 V	0.65 Vcc	—	Vcc	V
			2.7 V≦Vcc＜4.0 V	0.7 Vcc	—	Vcc	V
			1.8 V≦Vcc＜2.7 V	0.8 Vcc	—	Vcc	V
VIL	L入力電圧	CMOS入力以外		0	—	0.2 Vcc	V
		CMOS入力	4.0 V≦Vcc≦5.5 V	0	—	0.4 Vcc	V
			2.7 V≦Vcc＜4.0 V	0	—	0.3 Vcc	V
			1.8 V≦Vcc＜2.7 V	0	—	0.2 Vcc	V
IOH(sum)	H尖頭総出力電流	全端子のIOH(peak)の総和		—	—	-160	mA
IOH(sum)	H平均総出力電流	全端子のIOH(avg)の総和		—	—	-80	mA
IOH(peak)	H尖頭出力電流		駆動能力Low時	—	—	-10	mA
			駆動能力High時(注5)	—	—	-40	mA
IOH(avg)	H平均出力電流		駆動能力Low時	—	—	-5	mA
			駆動能力High時(注5)	—	—	-20	mA
IOL(sum)	L尖頭総出力電流	全端子のIOL(peak)の総和		—	—	160	mA
IOL(sum)	L平均総出力電流	全端子のIOL(avg)の総和		—	—	80	mA
IOL(peak)	L尖頭出力電流		駆動能力Low時	—	—	10	mA
			駆動能力High時(注5)	—	—	40	mA
IOL(avg)	L平均出力電流		駆動能力Low時	—	—	5	mA
			駆動能力High時(注5)	—	—	20	mA

LEDの定電流ドライブ

なぜ定電流ドライブが必要なのか？

個々のLEDにV_Fのバラツキがあり、電圧では正確に制御できない。
わずかな電圧差で大電流が流れて定格を超える。

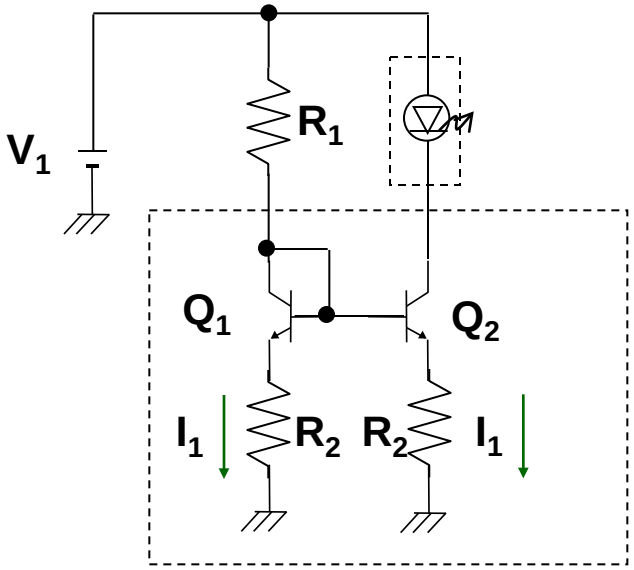
例えばある白色ダイオードの
V_Fと順方向電流は右表
最大定格電流は 25mA
100mV オーバでNG

3.2V	-	17mA
3.3V	-	39mA
3.4V	-	90mA

定電流回路の作り方

①定電圧電源と電流制限抵抗
②定電流ダイオード

①定電圧回路と電流制限抵抗（その1）



カレントミラー回路

定電流回路の一例

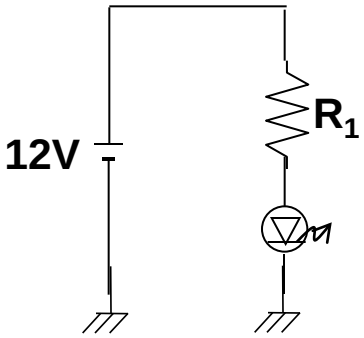
Q_2のコレクタ電圧値にかかわらず
コレクタ電流は一定値（I1）になる

$$I_1 = \frac{V_1 - V_{BE1}}{(R_1 + R_2)}$$

V_1 = 12V　　V_{BE1} = 0.7V
R_1 = 680Ω　R_2 = 100Ω
とすると、　I_1 = 14.5mAとなる

定電流ドライブ回路

①定電圧回路と電流制限抵抗（その2）



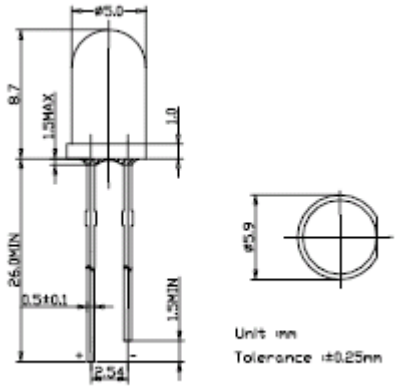
左図の回路でLEDに15mAの電流を流してみましょう
LEDは種類（メーカ）、色によって順方向電圧が異なります

色	型名	V_F@ 15mA	SPEC@20mA			最大定格 (Imax)
			min	typ.	max	
黄	OSYL5111A	1.95V	1.8V	2.0V	2.4V	50mA
赤	OSHR5111A	2.05V	1.8V	2.0V	2.4V	50mA
白	OSWT5111A	3.40V	-	3.6V	4.2V	25mA

このほか、 青（3.0V～3.6V）、緑（約2.3V）もあります

5mm Red LED

OSHR5111A-TU



▪Absolute Maximum Rating (Ta=25℃)

Item	Symbol	Value	Unit
DC Forward Current	I_F	50	mA
Pulse Forward Current*	I_{FP}	150	mA
Reverse Voltage	V_R	5	V
Power Dissipation	P_D	200	mW
Operating Temperature	Topr	-30 ~ +85	℃
Storage Temperature	Tstg	-30 ~ +100	℃
Lead Soldering Temperature	Tsol	260℃/5sec	-

*Pulse width Max.10ms Duty ratio max 1/10

▪Electrical -Optical Characteristics (Ta=25℃)

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
DC Forward Voltage	V_F	I_F=20mA	1.8	2.0	2.4	V
DC Reverse Current	I_R	V_R=5V	-	-	10	μA
Domi. Wavelength	λ_D	I_F=20mA	620	625	630	nm
Luminous Intensity	Iv	I_F=20mA	10000	12000	14400	mcd
50% Power Angle	$2\theta_{1/2}$	I_F=20mA	-	15	-	deg

$$R_1 = \frac{12V - V_F}{15mA}$$

計算値　実際の抵抗値

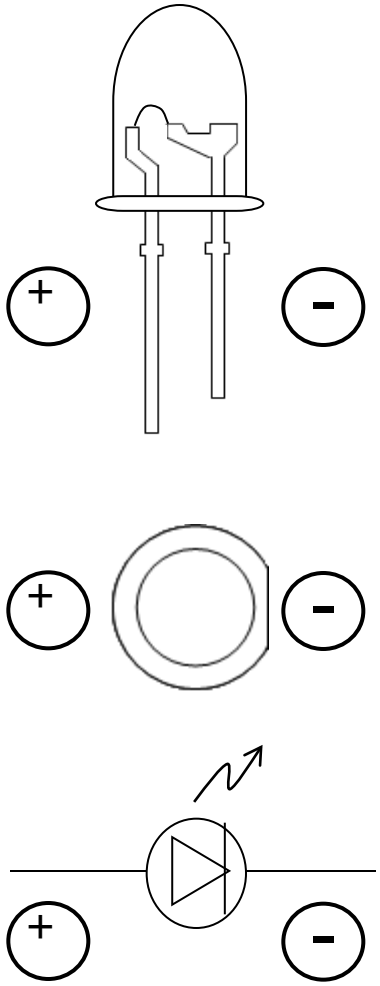
黄　$R_1 = \frac{12 - 1.95}{15mA} = 670\Omega \Rightarrow 680\Omega \quad 14.8mA$

赤　$R_1 = \frac{12 - 2.05}{15mA} = 663\Omega \Rightarrow 620\Omega \quad 16mA$

白　$R_1 = \frac{12 - 3.40}{15mA} = 573\Omega \Rightarrow 560\Omega \quad 15.3mA$

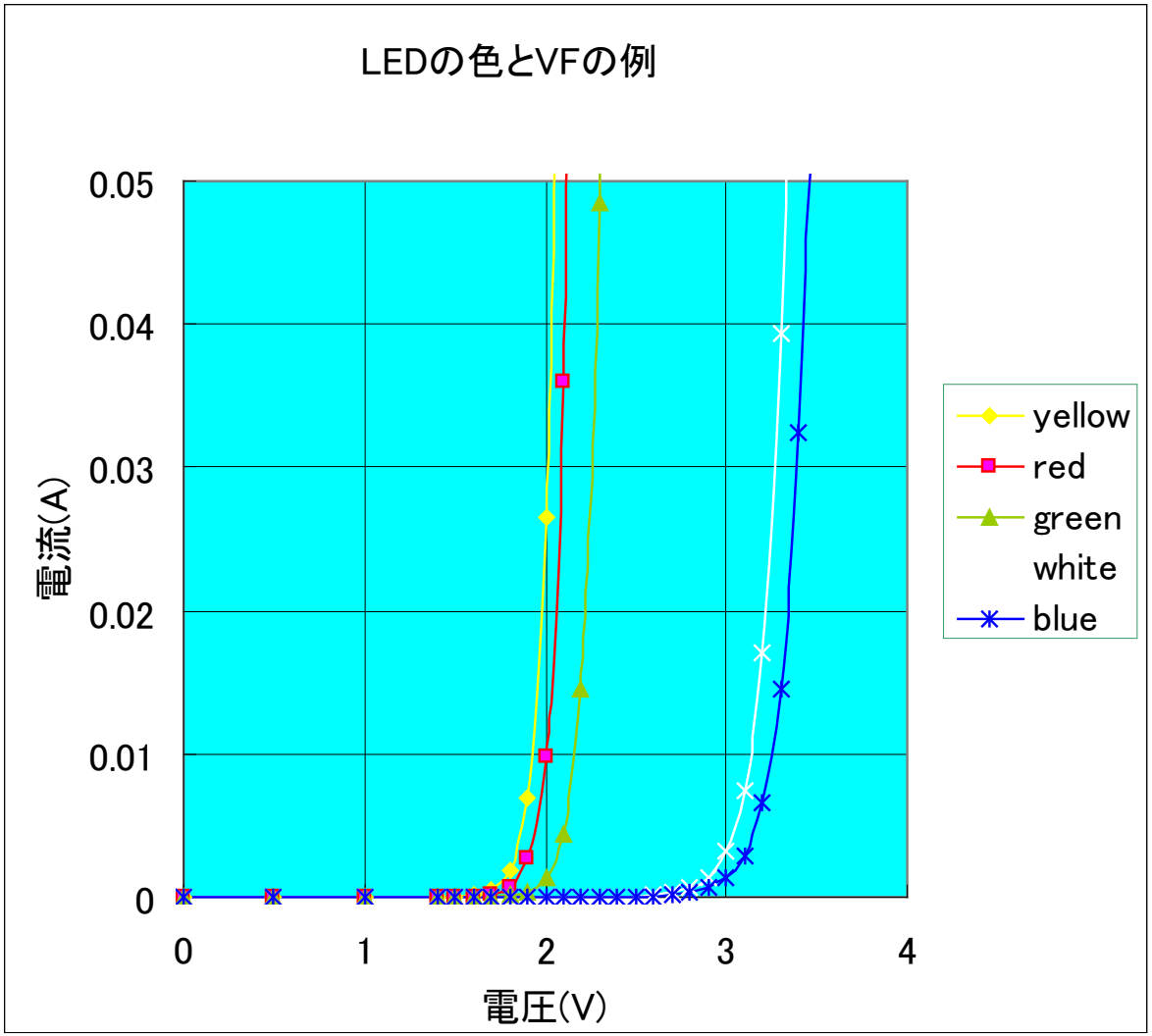
LEDの極性

超高輝度LED　OSYL5111A-TU（黄色）



LEDもダイオードなので極性がある。

LEDの逆電圧耐性は低いので注意！
（特に白と青は弱い　5V程度）



黄・赤・緑色は約2V、白・青色は3.3V程度が多い。 白・青は電池2本では苦しい。

PWM制御による輝度調整

PWM (Pulse Width Modulation) 信号による輝度調整

定電流量を変化させることにより輝度を変化させることはできます。

青色LEDではあまりわかりませんが、白色LEDでは定電流量を
変化させて輝度を変えると色温度が変わってしまうことが知られています。

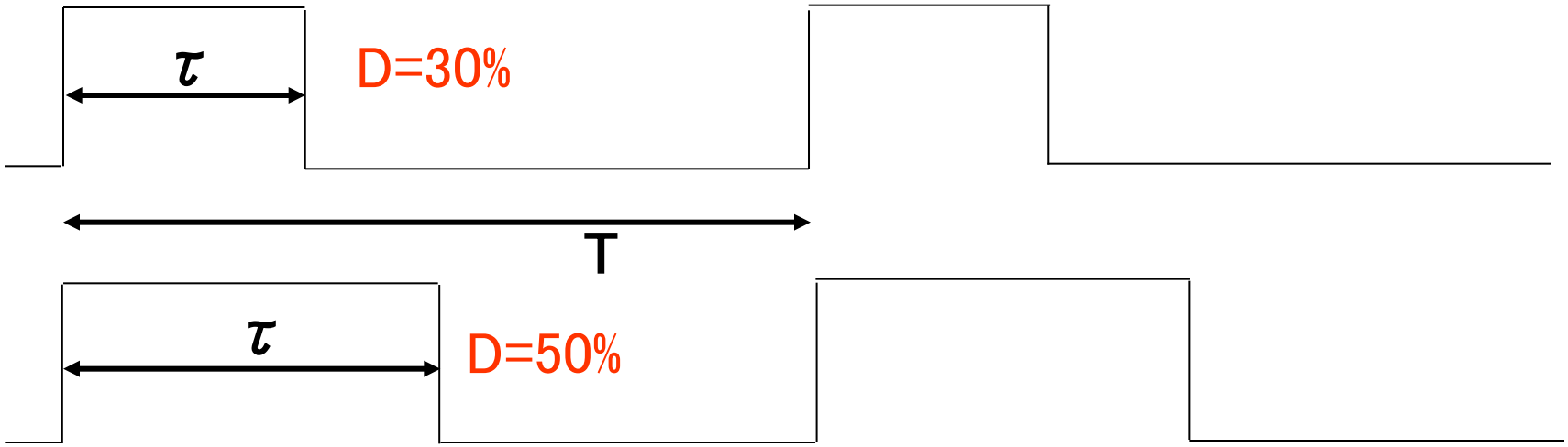
電流量を変化させずに、時間的に電流を流す時間（幅）を制御することにより輝度を調整する方法がPWM 制御方法です。

PWMとは

PWMは Pulse Width Modulationの略でパルス波のデューティ比を変化させて変調する変調方法です。
デューティ比とは周期的なパルス波を出したときの周期とパルス幅の比のことで以下の式で表されます。

$$D = \frac{\tau}{T}$$

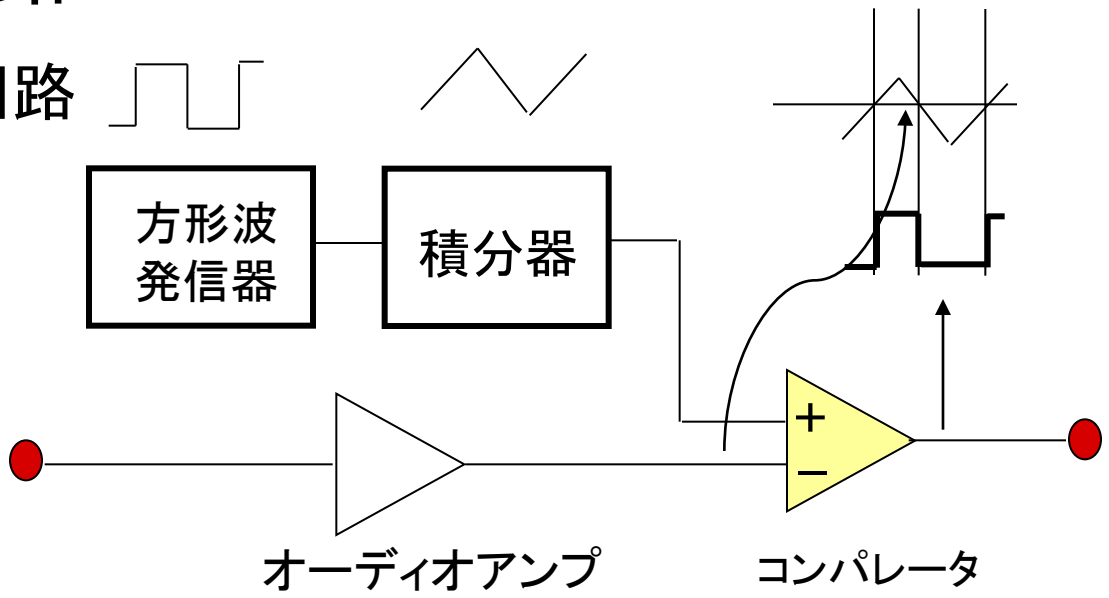
D：デューティ比
T：パルス周期
τ：パルス幅



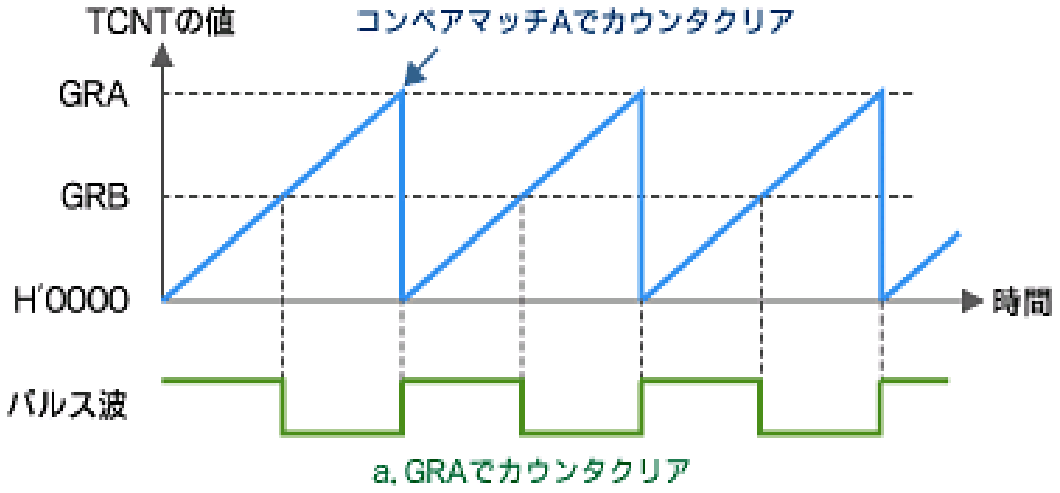
PWM発生回路

（1）ハードウェア回路

オーディオ信号の
PWM変換



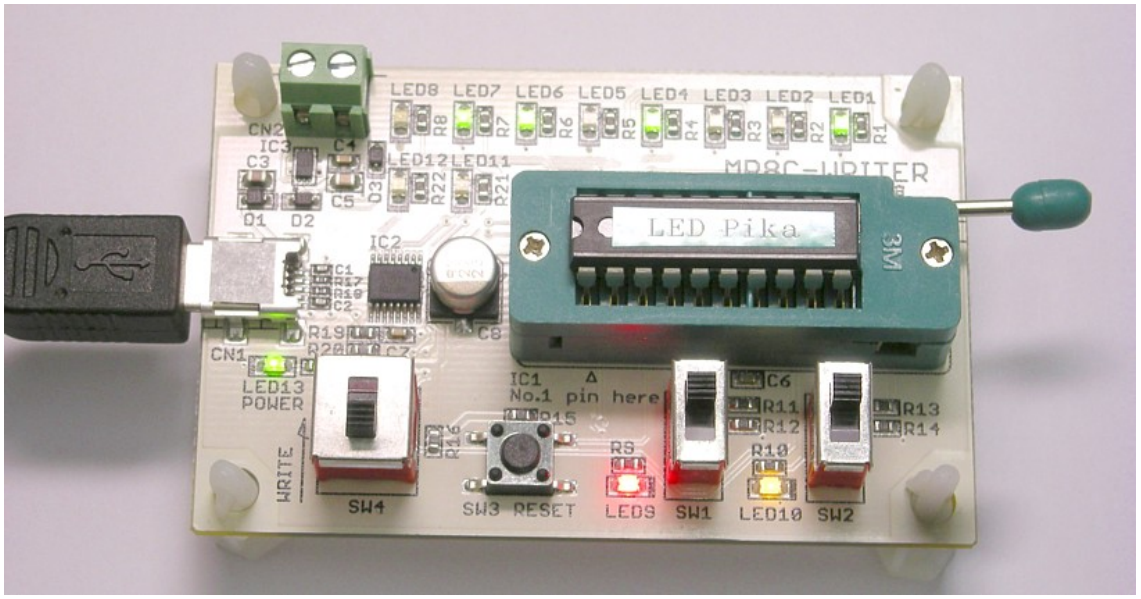
（2）マイコンではカウンタ使用で容易に発生可能



2. R8C/M12Aライタ＆LED基板の組み立て

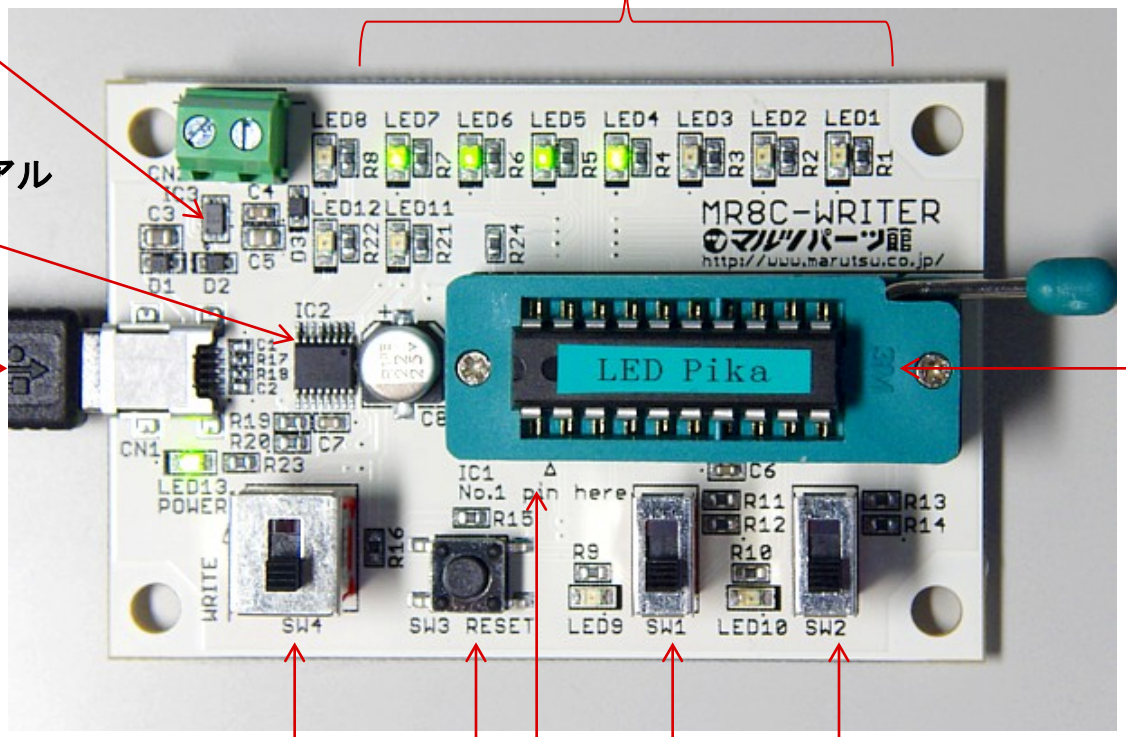
・基板組み立て
・回路説明
・動作チェック

基板の組み立てと動作確認



取り付ける部品はICソケットとスイッチ3個と電池コネクタ

R8C/M12A　ライタ＆LEDボード



ハンダ付け

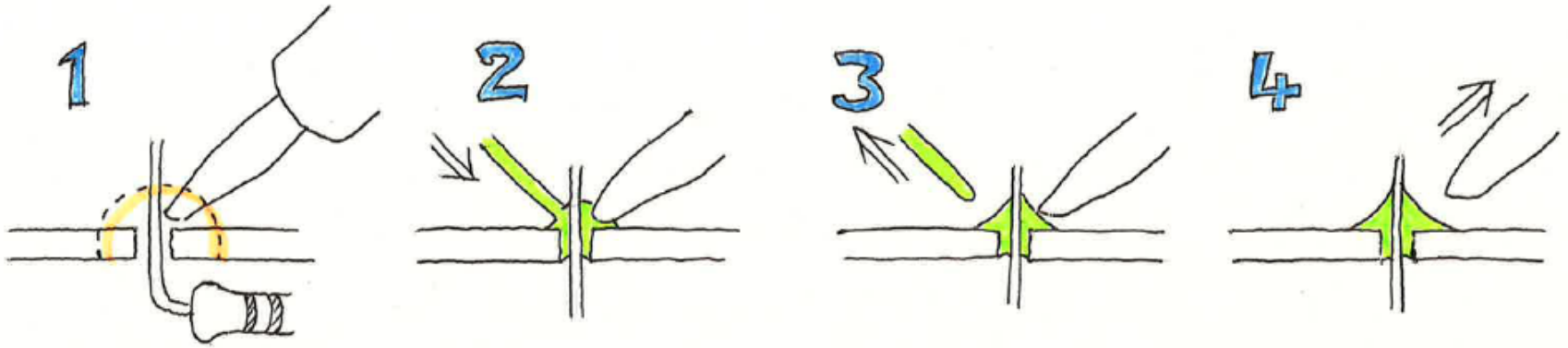
1. ハンダ付けする双方の部品を充分暖める
   双方とは、例えば抵抗の足と基板のパターン
   半田ごてを少し強く押し当てて熱を伝える

2. 半田を供給する
   ヌレと拡散 （きれいに溶けて広がる）

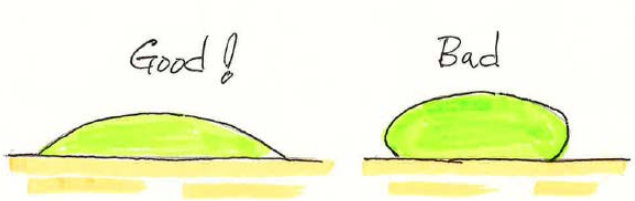
3. 半田を抜く

4. 半田ごてをすばやく離す

この間1，2，3！
約2-3秒



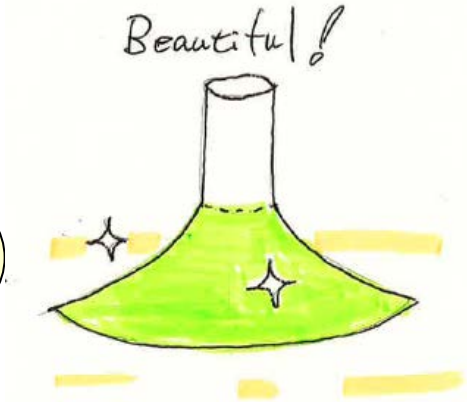
よいハンダ付けとは

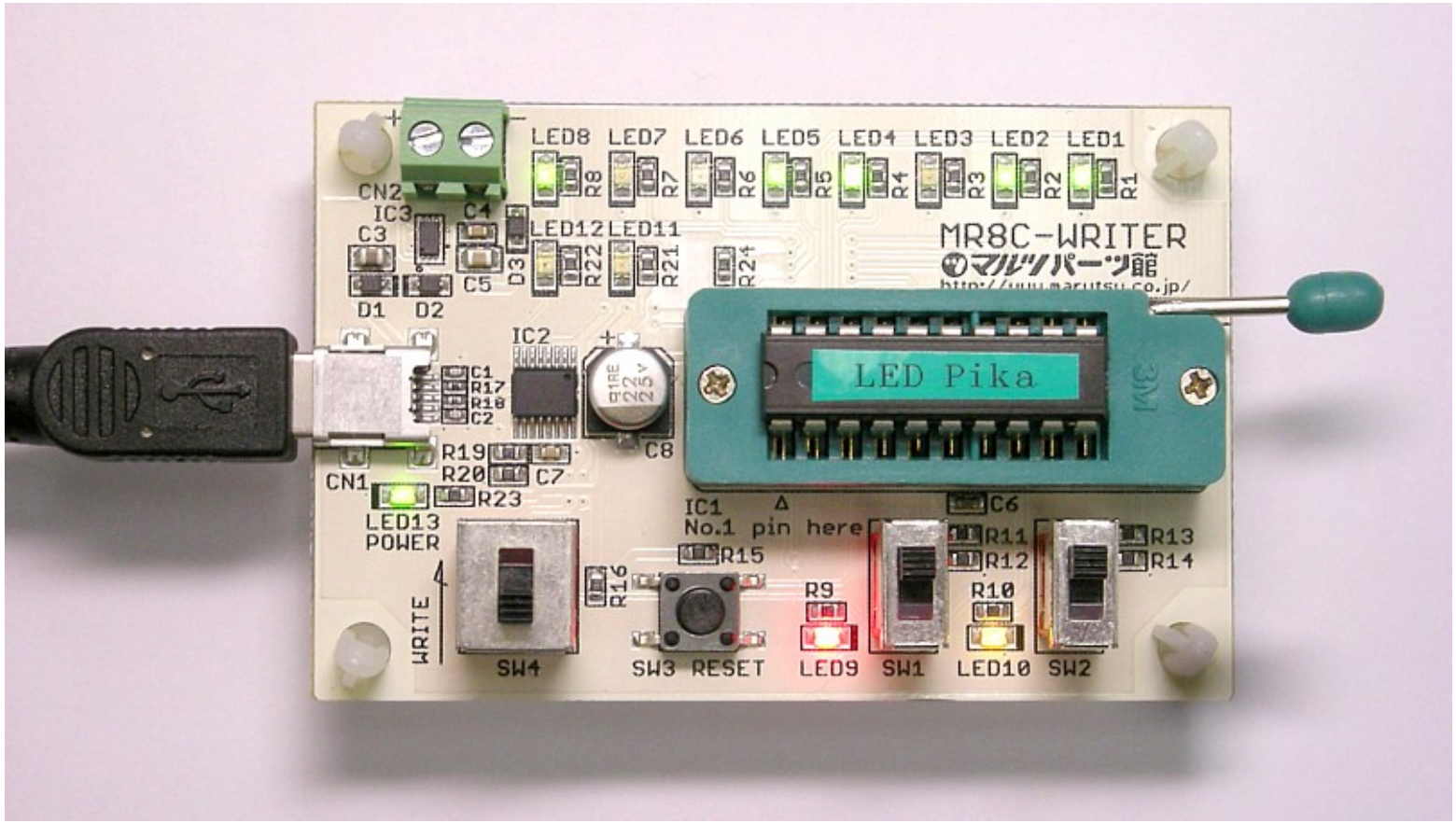


1. きれいに流れている、ボール状はダメ
2. ハンダ部分がきれいに輝いて滑らか
3. ハンダが厚すぎない、元の線の流れがわかる
4. われ、ヒビ、穴がない

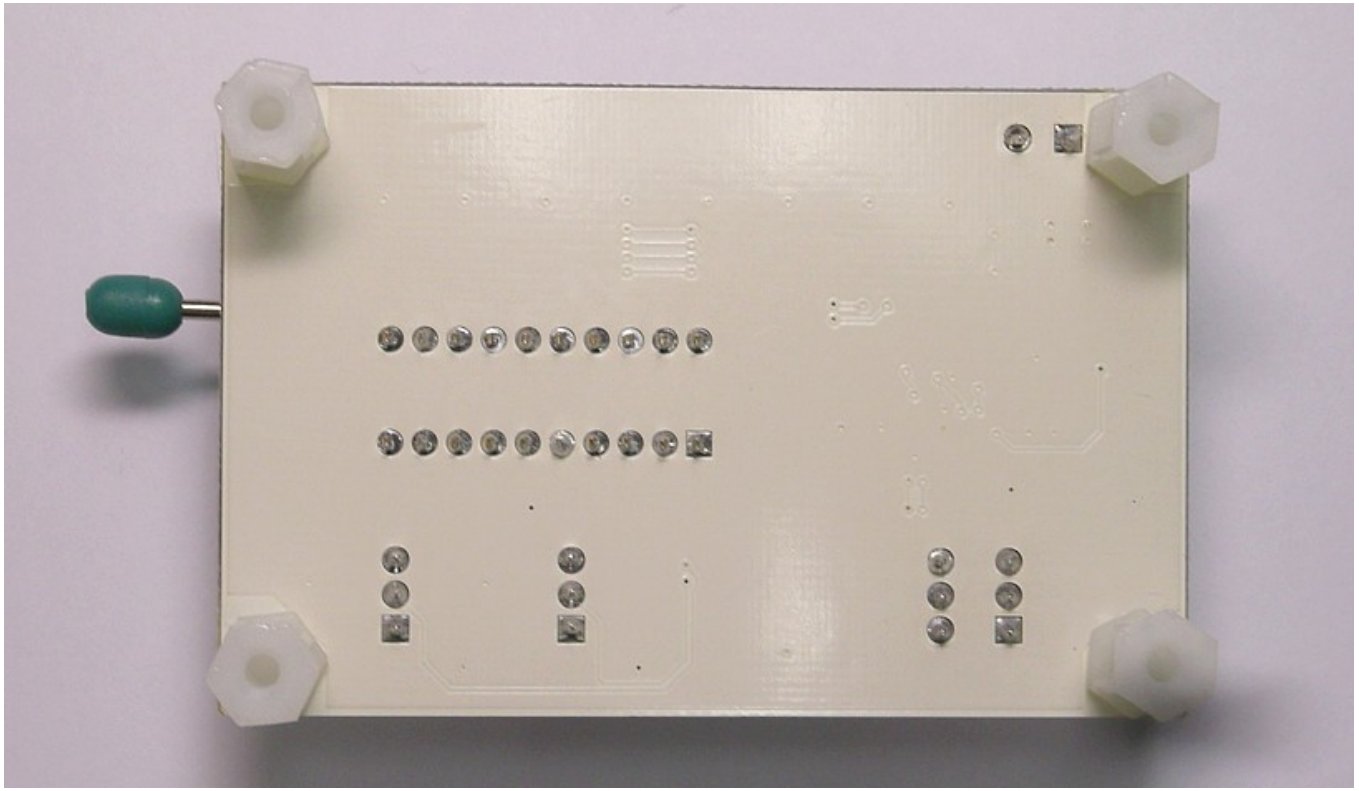
重要なポイント

1. 部品、パターンがきれい (さびてない)
2. 温度とタイミング
3. ハンダの量

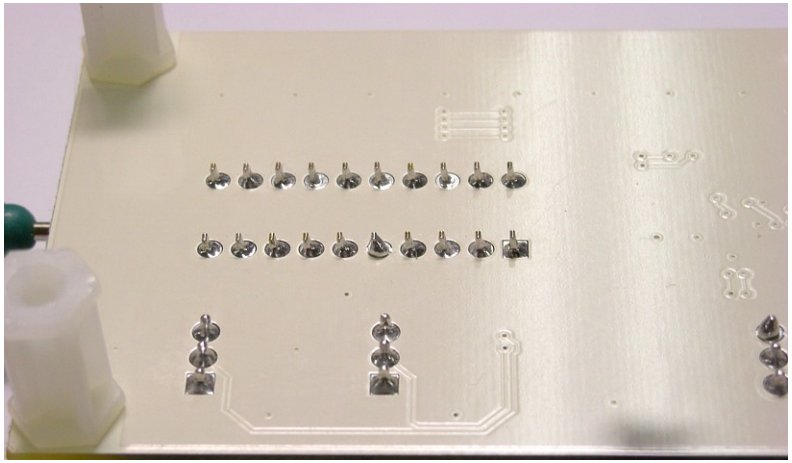




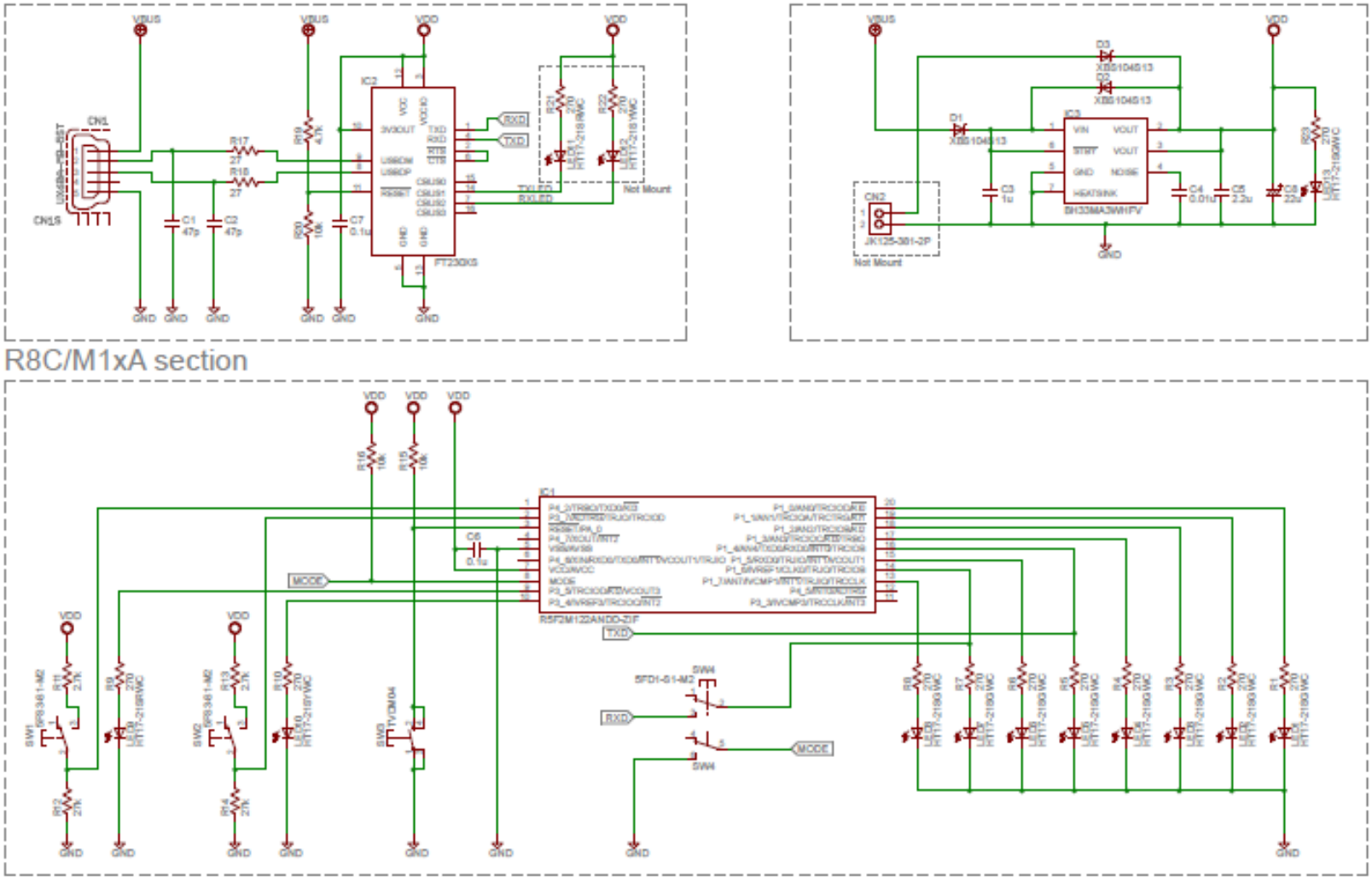
完成　表



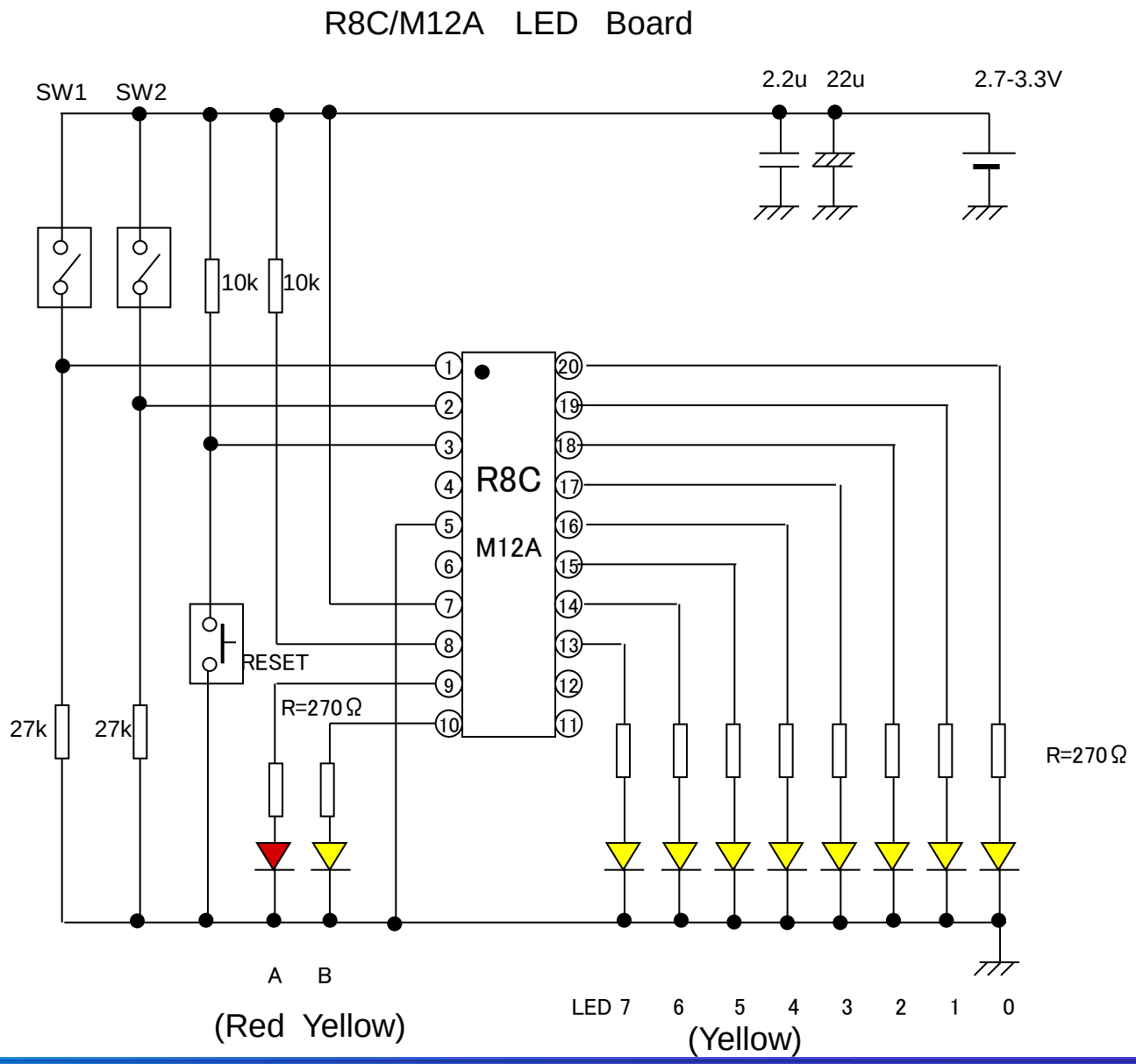
完成　裏



回路図



LEDフラッシャー主要回路図 （Play時）



ボードの動作確認

1. 書込み済のマイコン（M12A）を基板に実装し、SW4を下側（PLAY)にセット SW1とSW2を下側にセットしてUSBケーブルをPCに接続します。電源供給

2. USBコネクタ下のLEDが点灯し、8個のLEDの点灯が始まります。 リセットボタンを押すとLED1から順次左に約1.2秒毎点灯し、戻ってきて、2進数で255まで増加を続け、全点灯を5回繰り返します。

3. SW2をハイにしてリセットボタンを押すと、LED10オレンジが点灯し、動作速度が0.6秒ごとにアップします。

4. SW1をハイにして(SW2はロウ）リセットボタンを押すと、LED9赤が点灯し、動作速度が0.3秒にアップします。

5. SW1,SW2をハイにしてリセットボタンを押すと、LED9と10が点灯し、さらに高速に動作します。

6. LEDPikaTESTのソフトでは全体が約3倍の速度で動作します。

3. マイコン開発ツールの準備

・統合開発環境(HEWとコンパイラ)
・フラッシュ書込みソフト
・USBシリアルIFのデバイスドライバ

・LEDPika_TEST を書いてみよう！

マイコン開発に必要なもの

0. パソコン

1. 統合開発環境(HEW) 評価用無償版を使います　HEW V.4.09.01

2. コンパイラ （C言語を機械語に変換します）　M3T-NC30WA V.6.00

3. プログラム書き込みソフト　R8CWriter

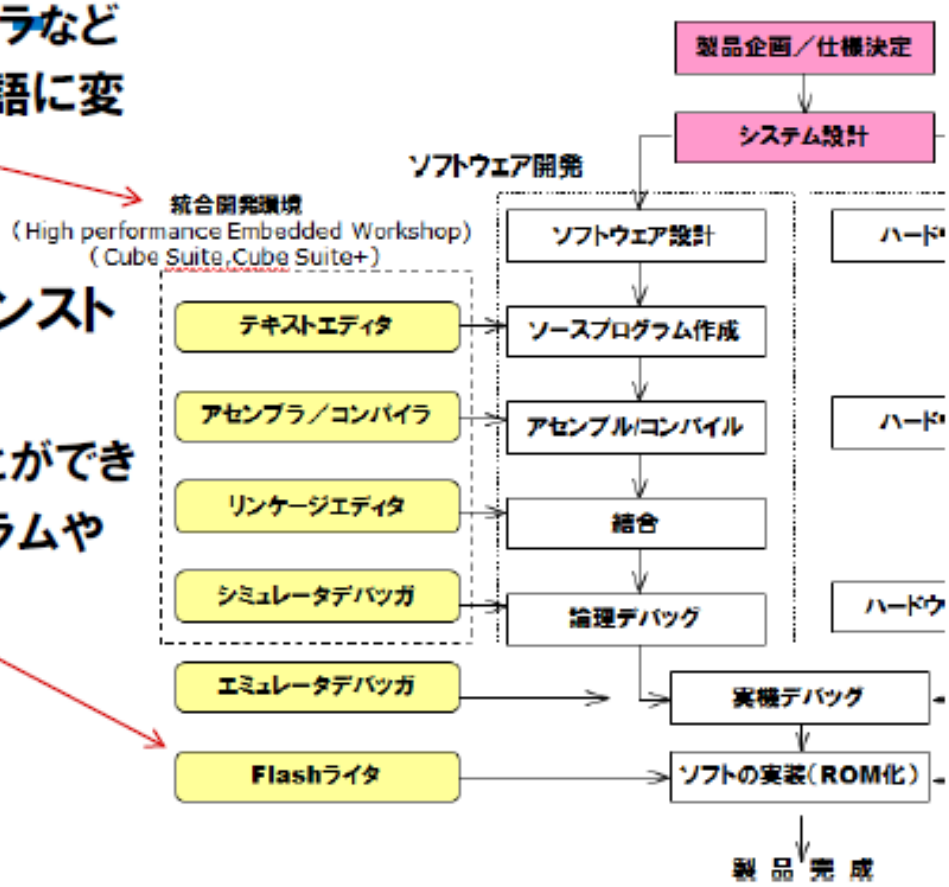
4. 書き込み基板と動作評価基板

5. 書き込み基板用IFのデバイスドライバ

HEW ： High-performance Embedded Workshop

開発環境(コンパイラ、マイコン書き込み)のインストール

- 統合開発環境(HEW)をインストール
  - プログラム入力用エディタ、コンパイラなどが含まれ、マイコンに書き込む機械語に変換できる
- FlashROMへの書き込みソフトをインストール
  - マイコン内蔵のROMに書き込むことができ、電源を切っても書き込んだプログラムやデータは消えない



コンパイラパッケージをダウンロード

- HEWはコンパイラパッケージに含まれています
  - 今回はUSBメモリで配布

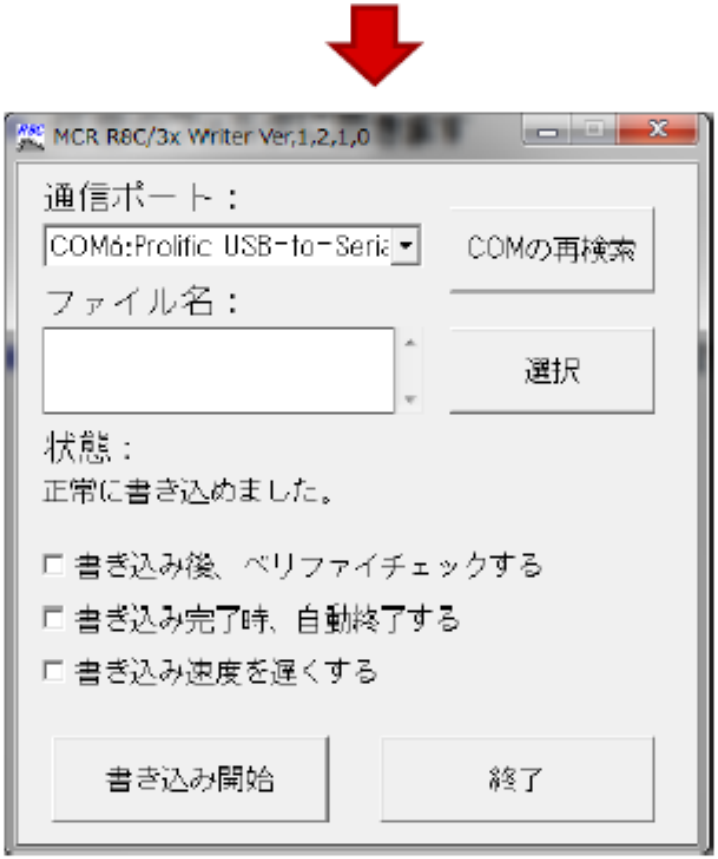
ダウンロードしたファイル

ダブルクリックで
インストール開始

FlashROMへの書き込みソフト

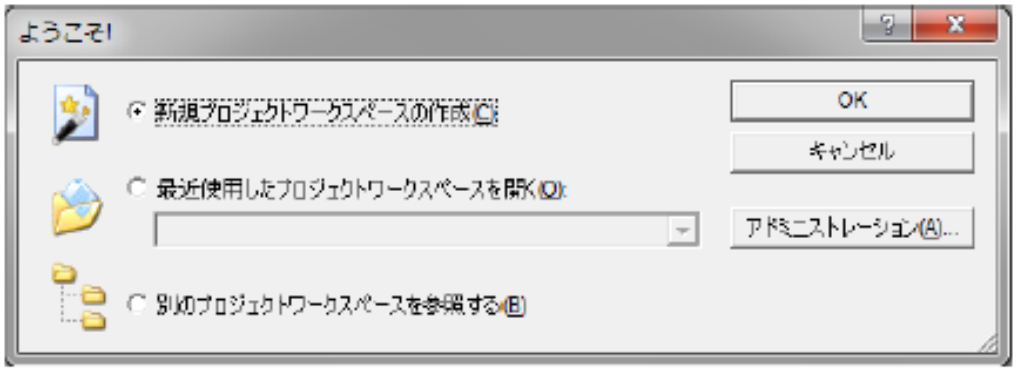
- 任意のフォルダに置きます

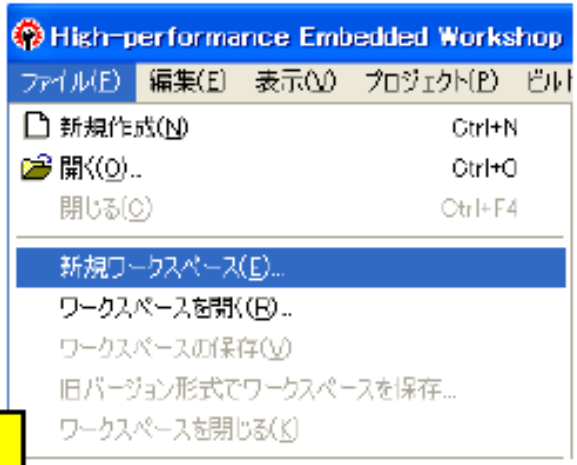
- ダブルクリックして起動できるか確認します



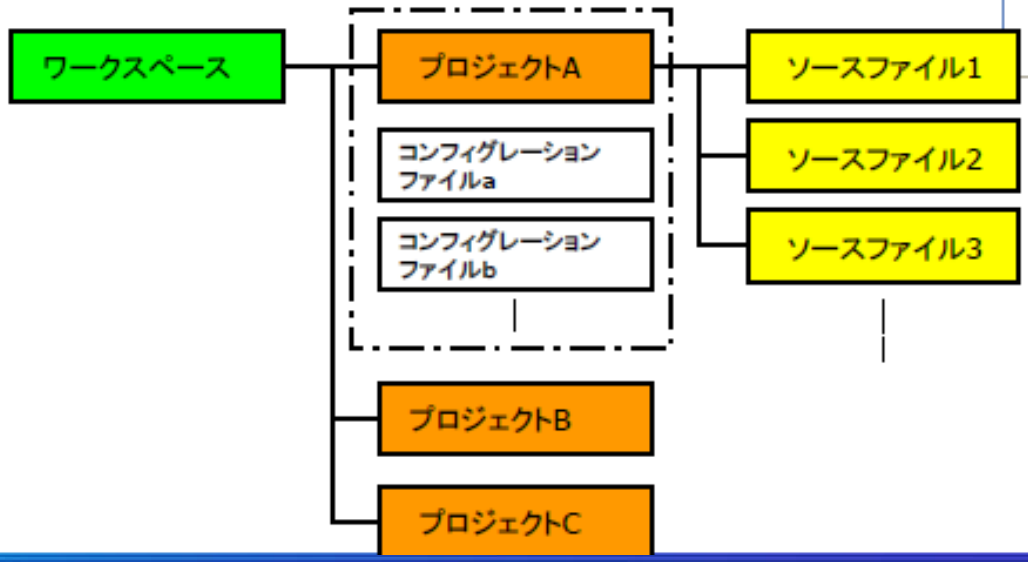
プログラム事始め （新規プロジェクトの作成の仕方を説明します。）

- HEWを起動します
  - 「ようこそ！」ダイアログで「新規プロジェクトワークスペースの作成」、または「ファイル」メニューの「新規ワークスペース」を選択





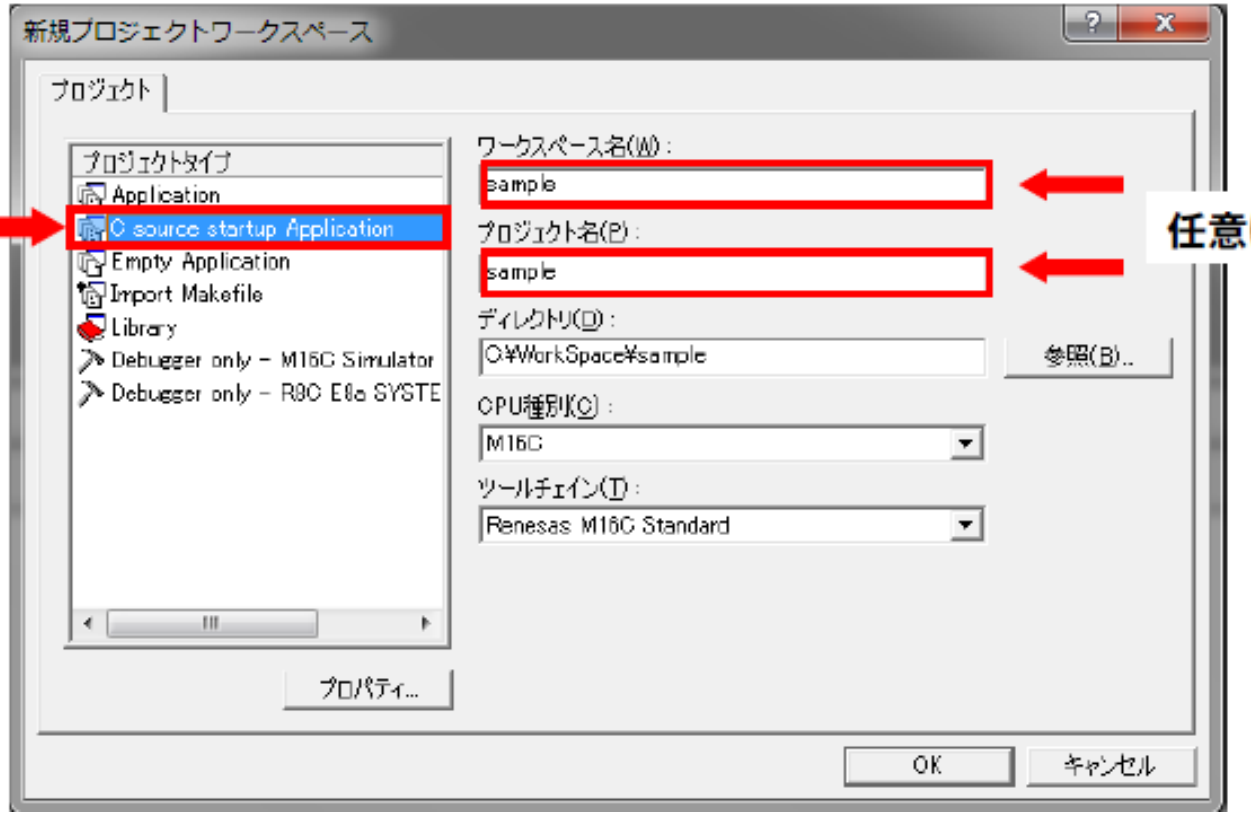
ワークスペース、プロジェクト、ソースファイルの関係



プロジェクトタイプの選択

- サンプルソースを生成するプロジェクトタイプは「C source startup Application」タイプです。
- ワークスペース名とプロジェクト名は任意ですが、今回は共に「sample」と入力します。

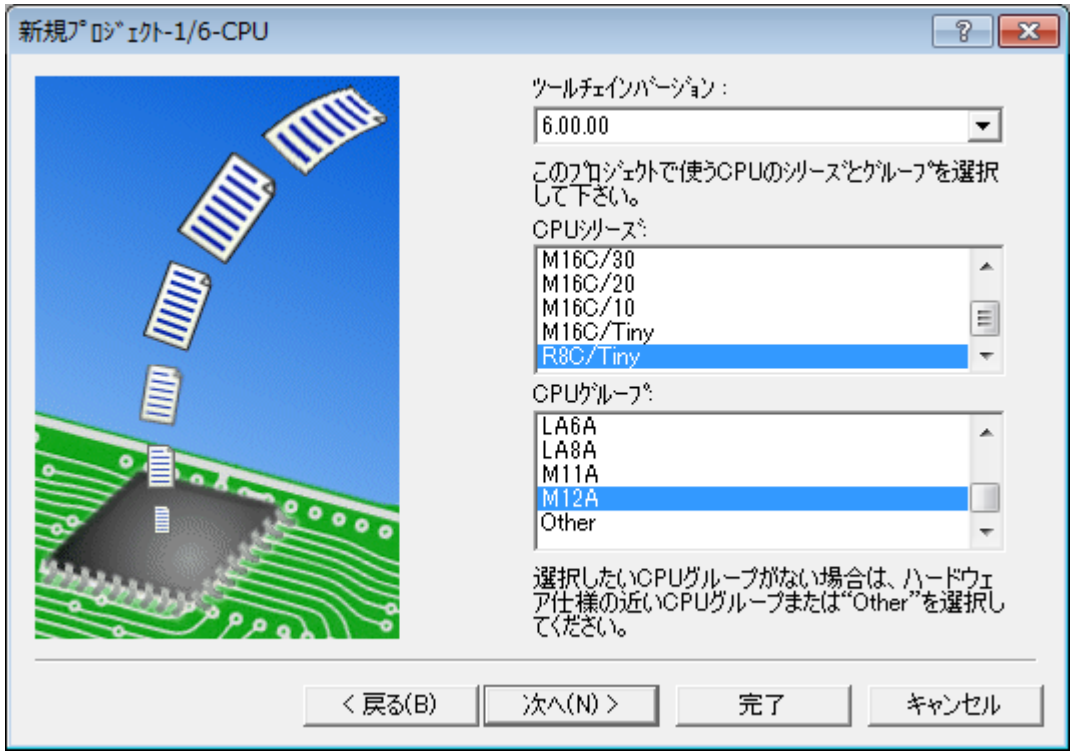
サンプルソースを生成する場合、C source startup Apllicationを選択



任意に選択可能

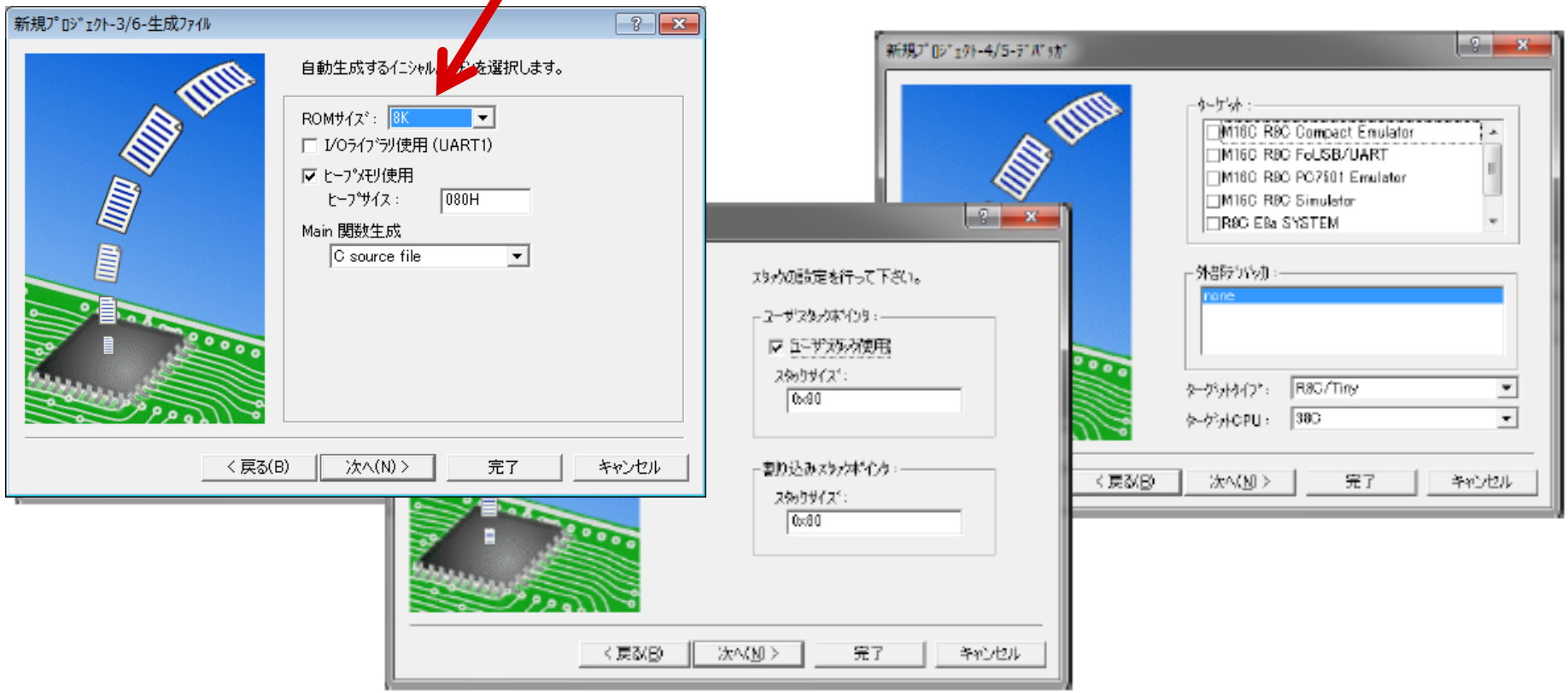
使用するCPUの選択

- 表示されたダイアログで使用するCPUシリーズとCPUタイプを選択します。
  CPUシリーズは「R8C/Tiny」、CPUタイプは「M12A」とします。



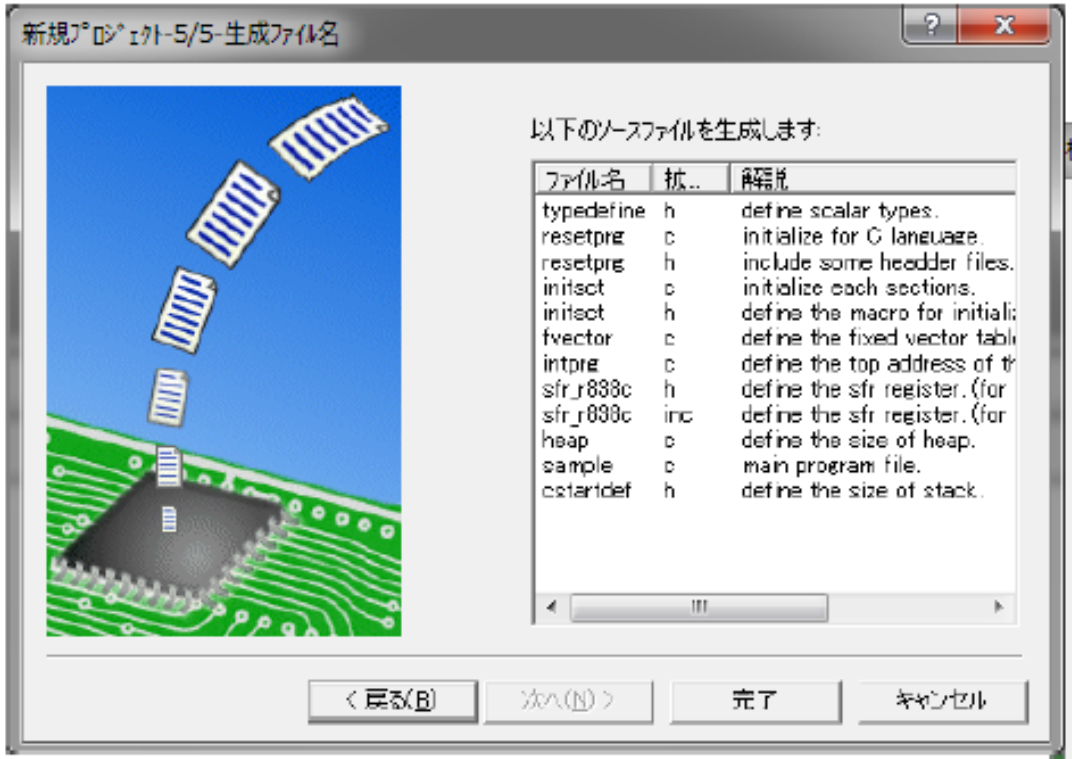
オプション設定

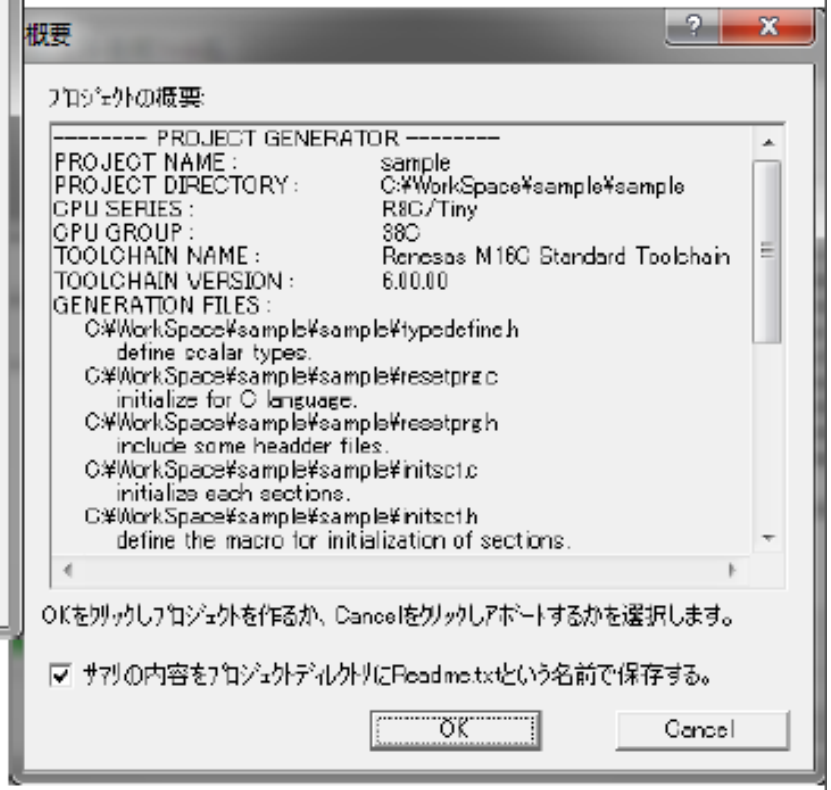
- 2/5-ROMサイズを　8KBに設定します。
- 3/5-設定を変更する必要はありません。
- 4/5-設定を変更するする必要はありません。



生成ファイル名

- 新規プロジェクトの設定は以上で終了です。
- 最終的に以下のソースファイルがHEWによって生成されます。





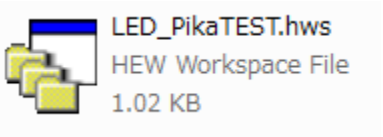
フラッシュメモリの書きこみとプログラム変更

フラッシュメモリ 書き込み手順

LED_PikaTESTフォルダを開く

↓

HEW ファイルをダブルクリック



↓

HEW 画面が開く
プログラムを変える
ビルド
エラーがないことをテェック

↓

PCとライタボードをつなぐ

↓

R8Cライタを立ち上げる

↓

ビルドファイルのアドレスセット

↓

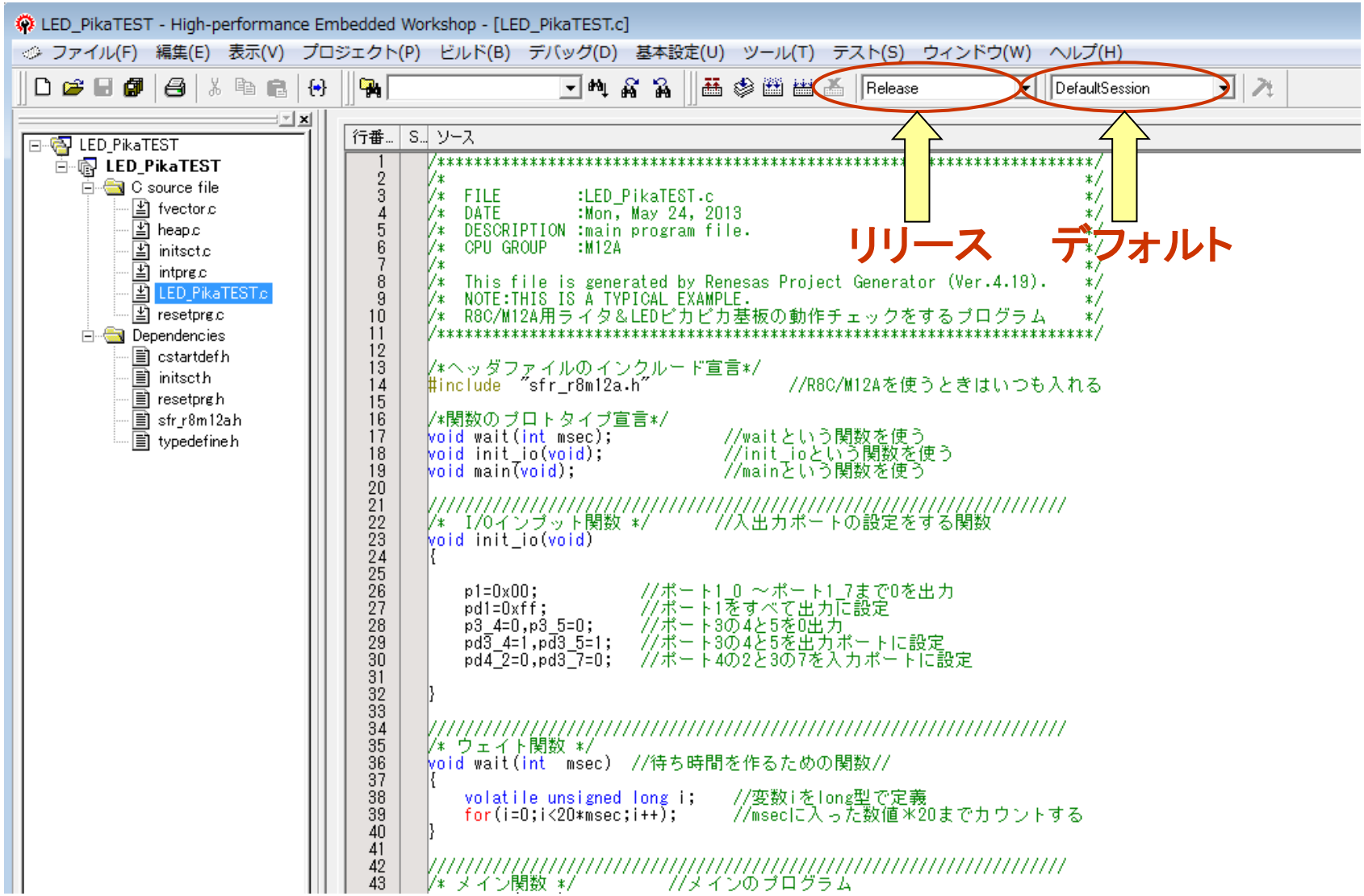
MCUをライタにセットしてライト

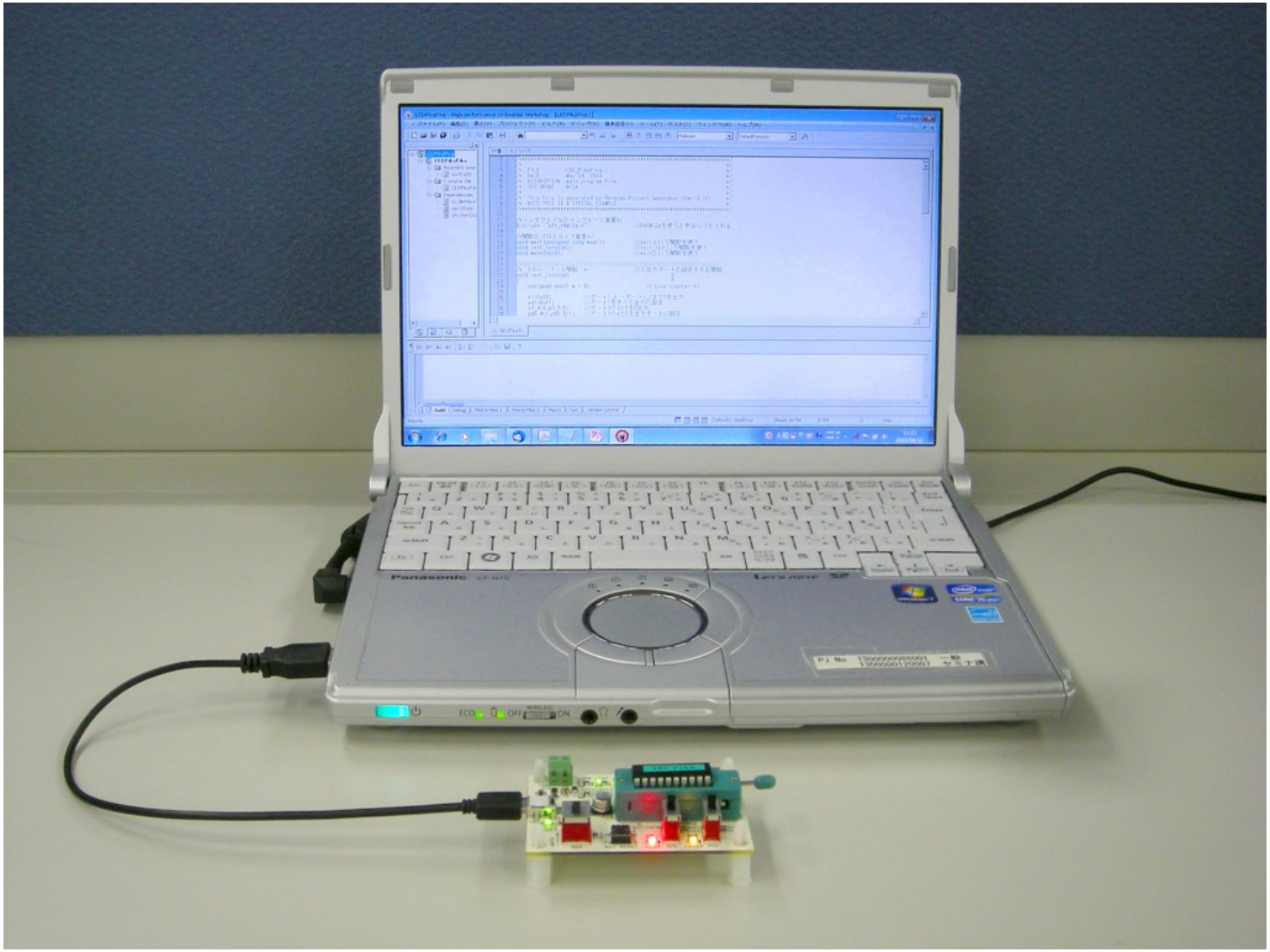
書込みSWをライト側に

↓

書込みSWを動作側にセットして Reset　SWを押す、
動作をチェック

HEWの設定

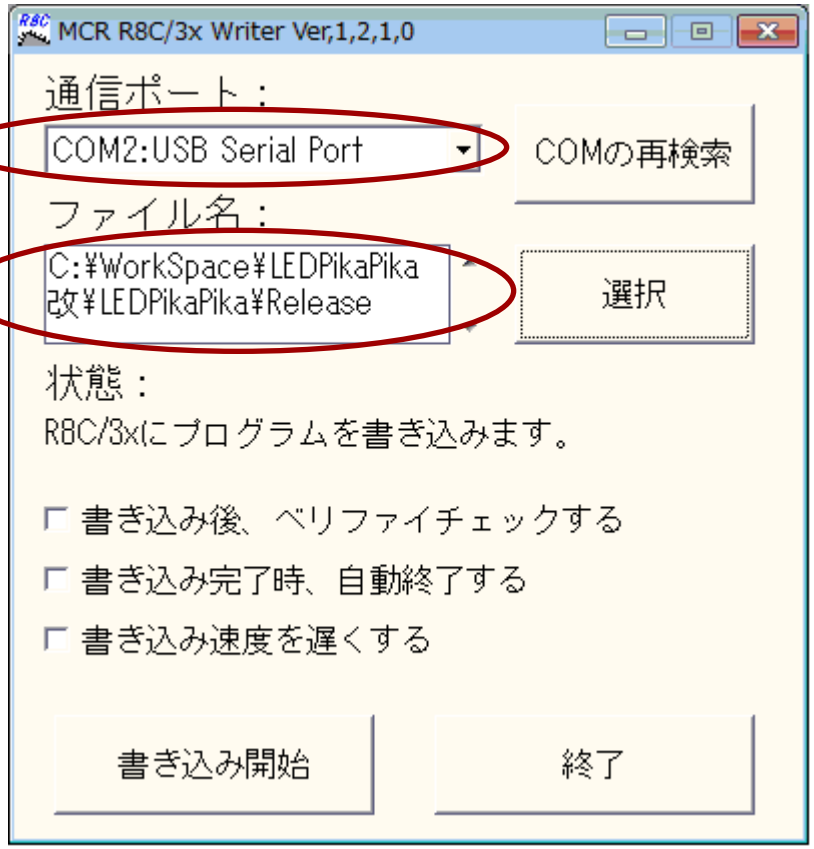




PCとライタボードの接続

R8Cライターの設定

環境によって変わります



4. LED　PikaPika プログラム（C言語）

・プログラムの解説
・プログラムの変更
・C言語と基本文法入門
・残り3つのプログラム

LED Pikaプログラムの説明

C言語のプログラムはたったこれだけです！

```
/*  NOTE:THIS IS A TYPICAL EXAMPLE.                                    */
/***********************************************************************/

/*ヘッダファイルのインクルード宣言*/
#include  "sfr_r8m12a.h"                  //R8CM12Aを使うときはいつも入れる

/*関数のプロトタイプ宣言*/
void wait(unsigned long msec);           //waitという関数を使う
void init_io(void);                      //init_ioという関数を使う
void main(void);                         //mainという関数を使う

/////////////////////////////////////////////////////////////////////
/*  I/Oインプット関数  */                //入出力ポートの設定をする関数
void init_io(void)
{
    unsigned short m = 0;                        /* Loop counter */

        p1=0x00;            //ポート1_0 ～ポート1_7まで0を出力
        pd1=0xff;           //ポート1をすべて出力に設定
        p3_4=0,p3_5=0;      //ポート3の4と5を0出力
        pd3_4=1,pd3_5=1;    //ポート3の4と5を出力ポートに設定
        pd4_2=0,pd3_7=0;    //ポート4の2と3の7を入力ポートに設定

}

/////////////////////////////////////////////////////////////////////
/*  ウェイト関数  */
void wait(unsigned long msec)  //待ち時間を作るための関数//
{
        volatile unsigned long i;            //変数iをlong型(32bit)で定義
        for(i=0;i<20*msec;i++);              //msecに入った数値*20までカウントする
}

/////////////////////////////////////////////////////////////////////
/* メイン関数  */                        //メインのプログラム
```

R8C/M12Aを使うときは付けるおまじない。Sfrです。

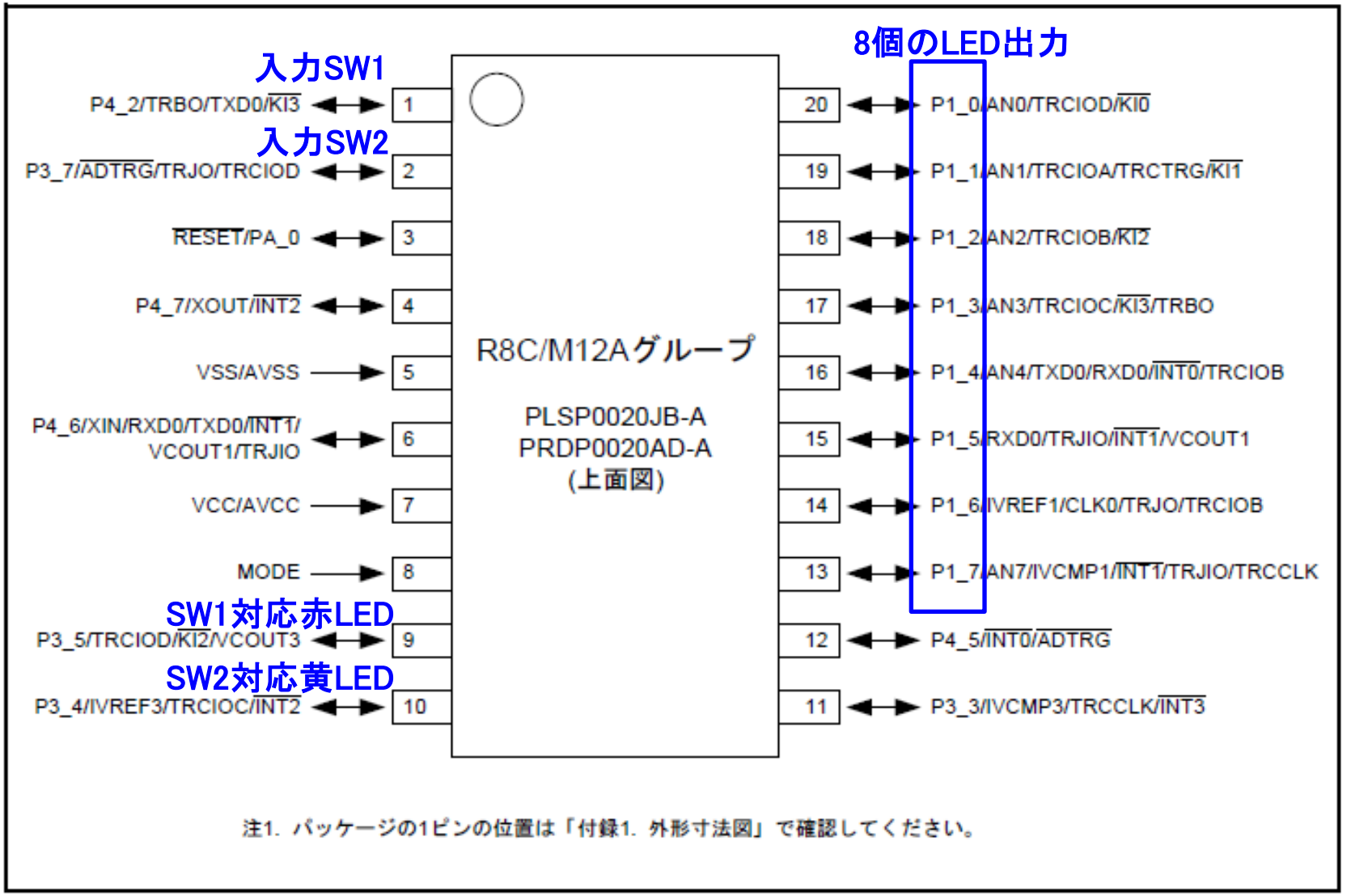
ウェイト(待ち時間)関数

ピンの入出力設定

メインプログラム

これ以外の指定していないピンは
初期値で入力ポートになっています。

クロックの設定をしていないので初期値
で125kHzの内部発振になっています。



注1. パッケージの1ピンの位置は「付録1. 外形寸法図」で確認してください。

図1.4　R8C/M12Aグループのピン配置図(上面図)

```
////////////////////////////////////////////////////////////////////
/* メイン関数  */                          //メインのプログラム
void main(void)
{
        int     i;                              //iという変数をint型(16bit)で定義
        char    LED;                            //LEDという変数をchar型(8bit)で定義
        unsigned long   n;                      //nという変数を符号なしlong型で定義

        init_io();                              //init_io()関数を実行
        p1=0x00;                                //ポート1にオール0を出力

        while(1){                               //無限ループ

    if      (p4_2==0 && p3_7==0) p3_4=0,p3_5=0,n=60;  //SW1=0,SW2=0ならLED10オフ、LED9オフ、n=60とする
    else if (p4_2==0 && p3_7==1) p3_4=1,p3_5=0,n=30;  //SW1=0,SW2=1ならLED10オン、LED9オフ、n=30とする
    else if (p4_2==1 && p3_7==0) p3_4=0,p3_5=1,n=15;  //SW1=1,SW2=0ならLED10オフ、LED9オン、n=15とする
    else                         p3_4=1,p3_5=1,n=3;   //上記以外ならLED10オン、LED9オン、n=3とする

                LED=0x01;                       //変数LEDを0x01にする
                for(i=0;i<8;i++){               //変数iを0から7まで繰り返す
                        p1=LED;                 //LEDの値をポート1に出力する
                        wait(n);                //変数nでwait関数を実行
                        LED=LED<<1;             //変数LEDの値を1ビット左にシフト
                }
                LED=0x80;                       //変数LEDを0x80にする
                for(i=0;i<8;i++){               //変数iを0から7まで繰り返す
                        p1=LED;                 //変数LEDの値をポート1に出力する
                        wait(n);                //変数nでwait関数を実行
                        LED=LED>>1;             //変数LEDの値を1ビット右にシフト
                }
                wait(n*2);                          //nの2倍時間待つ
                i=0;                                //i=0とする
                while(1){                           //無限ループ
                        p1=i;                       //変数iの値をポート1に出力する
                        wait(n);                    //nの時間待つ
                        if(i==255) break;           //もしiが255ならループからでる、そうでなければ続ける
                        i++;                        //iに1を足してiに代入する（i=i+1）
                }
                for(i=0;i<5;i++){                   //iを0から4まで5回繰り返す
                        p1=0x00;                    //ポート1をオール0にする（全消灯）
                        wait(n);                    //nの時間待つ
                        p1=0xff;                    //ポート1をオール1とする（全点灯）
                        wait(n*2);                  //nの2倍時間待つ
        }
    }
}
```

ここがご本尊のメイン関数

{ }のなかを繰り返します

一番右のLEDを点灯
nかぞえて
左に1個シフト
7回やったら

一番左のLEDを点灯
n数えて
右に1個シフトを
7回繰り返し

LED PikaPikaプログラムの変更

1. wait 関数の n の値を変えれば、待ち時間が変わります。
   LED PikaPika では　SW1、SW2の状態によって　60, 30, 15, 3としていますが
   LED PikaTEST では　テスト時間を短縮するため　20，8，4，2としています。
   60より大きくして勝手に点灯させておくと、なんだか癒されます。

   数字を変えてみましょう！

2. C言語のプログラムのコメントには // がついています。
   （// からその行の終わりまでは無視）
   また、/* から　*/ までも　コンパイル時にはプログラム以外として扱われます。
   （行をまたがっても、その間は無視）

   プログラムの繰り返しの部分に付けると、動作が変わります。

   //while(1){

   //{
   とすると繰り返しがなくなり、1回だけ実行して終了します。

待ち時間nの変更

100から1くらいまで

```
        while(1){                               //無限ループ

    if      (p4_2==0 && p3_7==0) p3_4=0,p3_5=0,n=60;  //SW1=0,SW2=0ならLED10オフ、LED9オフ、n=60とする
    else if (p4_2==0 && p3_7==1) p3_4=1,p3_5=0,n=30;  //SW1=0,SW2=1ならLED10オン、LED9オフ、n=30とする
    else if (p4_2==1 && p3_7==0) p3_4=0,p3_5=1,n=15;  //SW1=1,SW2=0ならLED10オフ、LED9オン、n=15とする
    else                         p3_4=1,p3_5=1,n=3;   //上記以外ならLED10オン、LED9オン、n=3とする

                LED=0x01;                       //変数LEDを0x01にする
                for(i=0;i<8;i++){               //変数iを0から7まで繰り返す
                        p1=LED;                 //LEDの値をポート1に出力する
                        wait(n);                //変数nでwait関数を実行
                        LED=LED<<1;             //変数LEDの値を1ビット左にシフト
                }
                LED=0x80;                       //変数LEDを0x80にする
                for(i=0;i<8;i++){               //変数iを0から7まで繰り返す
                        p1=LED;                 //変数LEDの値をポート1に出力する
                        wait(n);                //変数nでwait関数を実行
                        LED=LED>>1;             //変数LEDの値を1ビット右にシフト
                }
                wait(n*2);                          //nの2倍時間待つ
                i=0;                                //i=0とする
                while(1){                           //無限ループ
                        p1=i;                       //変数iの値をポート1に出力する
                        wait(n);                    //nの時間待つ
                        if(i==255) break;           //もしiが255ならループからでる、そうでなければ続ける
                        i++;                        //iに1を足してiに代入する（i=i+1）
                }
                for(i=0;i<5;i++){                   //iを0から4まで5回繰り返す
                        p1=0x00;                    //ポート1をオール0にする（全消灯）
                        wait(n);                    //nの時間待つ
                        p1=0xff;                    //ポート1をオール1とする（全点灯）
                        wait(n*2);                  //nの2倍時間待つ
        }
      }

}
```

プログラムの変更(2)

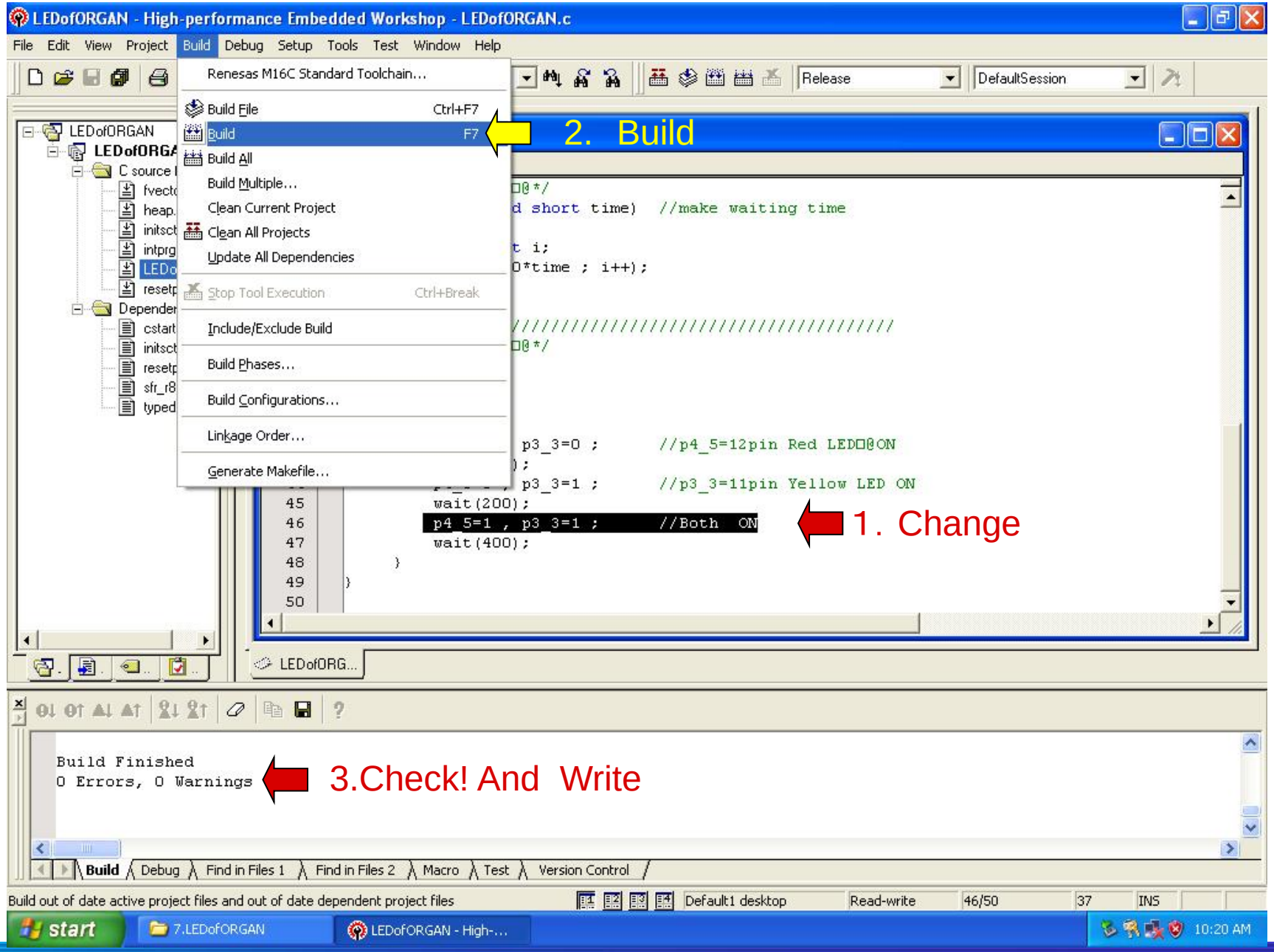
```
    //  while(1){                           //無限ループ

     if       (p4_2==0 && p3_7==0) p3_4=0,p3_5=0,n=60;  //SW1=0,SW2=0ならLED10オフ、LED9オフ、n=60とする
     else if  (p4_2==0 && p3_7==1) p3_4=1,p3_5=0,n=30;  //SW1=0,SW2=1ならLED10オン、LED9オフ、n=30とする
     else if  (p4_2==1 && p3_7==0) p3_4=0,p3_5=1,n=15;  //SW1=1,SW2=0ならLED10オフ、LED9オン、n=15とする
     else                          p3_4=1,p3_5=1,n=3;   //上記以外ならLED10オン、LED9オン、n=3とする

                 LED=0x01;                       //変数LEDを0x01にする
                 for(i=0;i<8;i++){               //変数iを0から7まで繰り返す
                         p1=LED;                 //LEDの値をポート1に出力する
                         wait(n);                //変数nでwait関数を実行
                         LED=LED<<1;             //変数LEDの値を1ビット左にシフト
                 }
                 LED=0x80;                       //変数LEDを0x80にする
                 for(i=0;i<8;i++){               //変数iを0から7まで繰り返す
                         p1=LED;                 //変数LEDの値をポート1に出力する
                         wait(n);                //変数nでwait関数を実行
                         LED=LED>>1;             //変数LEDの値を1ビット右にシフト
                 }
                 wait(n*2);                          //nの2倍時間待つ
       /*        i=0;                                //i=0とする
                 while(1){                           //無限ループ
                         p1=i;                       //変数iの値をポート1に出力する
                         wait(n);                    //nの時間待つ
                         if(i==255) break;           //もしiが255ならループからでる、そうでなければ続ける
                         i++;                        //iに1を足してiに代入する (i=i+1)
                 }  */
                 for(i=0;i<5;i++){                   //iを0から4まで5回繰り返す
                         p1=0x00;                    //ポート1をオール0にする (全消灯)
                         wait(n);                    //nの時間待つ
                         p1=0xff;                    //ポート1をオール1とする (全点灯)
                         wait(n*2);                  //nの2倍時間待つ
         }
     //}
}
```

プログラムの変更(3)

```
    while(1){                                   //無限ループ

 if       (p4_2==0 && p3_7==0) p3_4=0,p3_5=0,n=60;  //SW1=0,SW2=0ならLED10オフ、LED9オフ、n=60とする
 else if  (p4_2==0 && p3_7==1) p3_4=1,p3_5=0,n=30;  //SW1=0,SW2=1ならLED10オン、LED9オフ、n=30とする
 else if  (p4_2==1 && p3_7==0) p3_4=0,p3_5=1,n=15;  //SW1=1,SW2=0ならLED10オフ、LED9オン、n=15とする
 else                          p3_4=1,p3_5=1,n=3;   //上記以外ならLED10オン、LED9オン、n=3とする

            LED=0x01;                       //変数LEDを0x01にする
            for(i=0;i<8;i++){               //変数iを0から7まで繰り返す
                    p1=LED;                 //LEDの値をポート1に出力する
                    wait(n);                //変数nでwait関数を実行
                    LED=LED<<1;             //変数LEDの値を1ビット左にシフト
            }
            LED=0x80;                       //変数LEDを0x80にする
            for(i=0;i<8;i++){               //変数iを0から7まで繰り返す
                    p1=LED;                 //変数LEDの値をポート1に出力する
                    wait(n);                //変数nでwait関数を実行
                    LED=LED>>1;             //変数LEDの値を1ビット右にシフト
            }
            wait(n*2);                          //nの2倍時間待つ
            i=0;                                //i=0とする
            while(1){                           //無限ループ
                    p1=i;                       //変数iの値をポート1に出力する
                    wait(n);                    //nの時間待つ
                    if(i==255) break;           //もしiが255ならループからでる、そうでなければ続ける
                    i++;                        //iに1を足してiに代入する (i=i+1)
            }
            for(i=0;i<5;i++){                   //iを0から4まで5回繰り返す
                    p1=0x00;                    //ポート1をオール0にする (全消灯)
                    wait(n);                    //nの時間待つ
                    p1=0xff;                    //ポート1をオール1とする (全点灯)
                    wait(n*2);                  //nの2倍時間待つ
      }
     }

}
```

i=i+4

プログラムを変更したら、ビルドしてエラーのないことを確認して書き込みます。



練習1. 左4個点灯、右4個消灯　と　左4個消灯、右4個点灯を1秒ごとに繰り返す。

練習2. 次のパターンで点灯するプログラムを作ってみましょう。(1:点灯、0:消灯)

①「1000 0001」を0.5秒点灯
②「1100 0011」を0.5秒点灯
③「1110 0111」を0.5秒点灯
④「1111 1111」を0.5秒点灯
⑤「0000 0000」を1秒点灯
⑥①に戻って繰り返す

LEDPikaPikaの wait 関数で　n=30で約0.6秒です。

C言語と基本文法入門

C言語のスタイル

- プログラムは関数の集まり。
- 関数は文の集まり。
- 文の終わりにセミコロン(;)をつける。
- 慣用的に小文字を用いて書く。
- 複合文は { } で囲み、ブロック化する。
- インデント(字下げ)で見やすくする。
- コメント(注釈)は /* と */ で囲む。

```
/*コメント*/
/*関数の原型(プロトタイプ)宣言*/
int main(void);
int sub(int);

/*グローバル変数の宣言*/
int a;

/*関数の始まり*/
main()
{
    int b, c;           /*ローカル変数の宣言*/

    c = sub(b);         /*sub関数の呼び出し*/
    a = c + 128;        /*演算*/
}

/*関数の始まり*/
int sub(int x)
{
    return(b * 100 - 20);
}
/*ファイルの終了*/
```

定数(値を変えないもの)

数値定数	整数定数	8進数	先頭に「0」をつけて表記(077など)
		10進数	通常の10進数と同じ表記
		16進数	先頭に「0x」をつけて表記(0x41など)
	浮動小数点定数	小数点以下が扱える数(16.25 や 1.0e-3 など)	
文字定数	1文字のこと。'A' や 'B' のように ' ' で囲んで表記		
文字列定数	複数文字のこと。"ABC" や "computer"のように " "で囲んで表記		

変数(任意の値をとりうるもの)

型指定	データ型	バイト幅	扱える数値の範囲
char	文字型	1	-128～127
short	短長整数型	2(2バイト以上)	-32768～32767
int	整数型	2または4(short以上)	
long	倍長整数型	4	-2147483648～2147483647
float	単精度浮動小数点型	4	3.4E-38～3.4E+38
double	倍精度浮動小数点型	8	1.7E-308～1.7E+308

整数型には符号が付かないunsignedがある

変数の名付けルール

先頭の文字は英字(a～z、A～Z)または下線(_)でなければならない。
2文字目以降は英字、下線、数字である。
大文字と小文字を区別する。
先頭から最低31文字までが有効である。
予約語(C言語の文法上で使われる語)と同じ綴り(do、for など)は使用できないが、識別名の一部に入るのは可(dot、form など)。

予約語

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

演算子

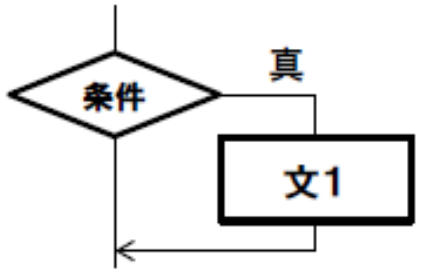
演算	演算子	例	意味
代入	=	a = b	a に b を代入する
加算	+	a + b	a に b を加える
減算	-	a - b	a から b を引く
乗算	*	a * b	a に b をかける
除算	/	a / b	a を b で割る
剰余算	%	a % b	a を b で割った余り
インクリメント	++	a++	a に 1 を加える(後置演算)
		++a	a に 1 を加える(前置演算)
デクリメント	--	a--	a から 1 を引く(後置演算)
		--a	a から 1 を引く(前置演算)
ビット毎の演算	&	a & b	a と b をビット毎にAND
	|	a | b	a と b をビット毎にOR
	^	a ^ b	a と b をビット毎にEOR
論理演算	!	!a	a の真・偽を反転
	&&	a && b	a と b の真・偽をAND
	||	a || b	a と b の真・偽をOR
シフト	>>	a >> b	a を b ビット数分右にシフト
	<<	a << b	a を b ビット数分左にシフト

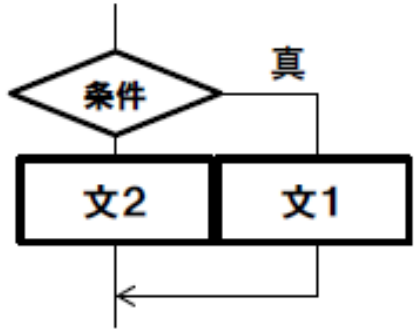
演算子	機能
+=	足して代入
-=	引いて代入
*=	掛けて代入
/=	割って代入
%=	余って代入
<<=	左シフトして代入
>>=	右シフトして代入
&=	ANDして代入
|=	ORして代入
^=	NOTして代入

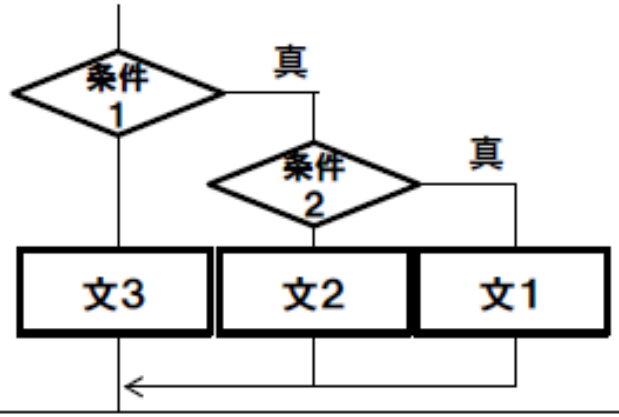
条件判断(if else文)

ある条件を満たすか満たさないかにより処理を分ける

フローチャート	書式

	if (条件式)
	{
	文1;
	}

	if (条件式)
	{
	文1;
	}
	else
	{
	文2;
	}

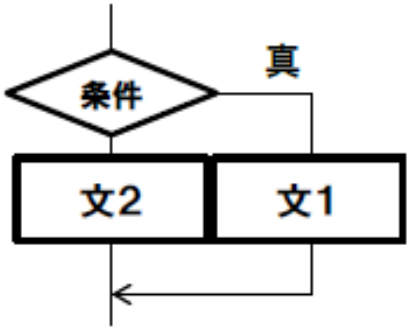
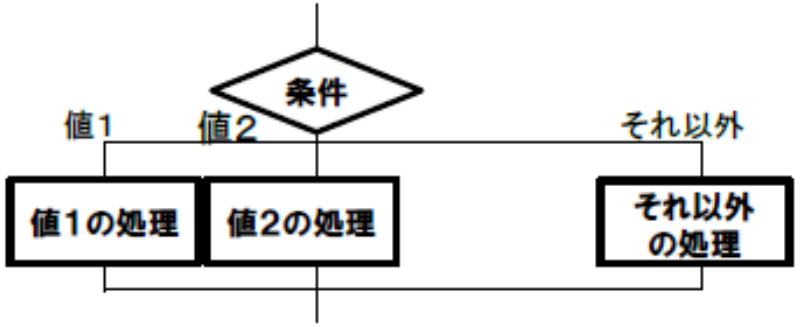
	if (条件式1)
	{
	if (条件式2)
	{ 文1; }
	else
	{ 文2; }
	}
	else
	{ 文3; }

関係演算子と論理演算子

関係演算子	
a<b	aはbよりも小さい
a>b	aはbよりも大きい
a<=b	aはb以下
a>=b	aはb以上
a==b	aはbと等しい
a!=b	aはbと等しくない

論理演算子	
a&&b	aとbが真なら真、でなければ偽になる(「しかも」という意味になる)
a||b	a, bのどちらかが真なら真、でなければ偽になる(「または」という意味になる)
!a	真偽が反転する(「...ではなかったら」という意味になる)

switch case文

	if文	switch case文
フローチャート		

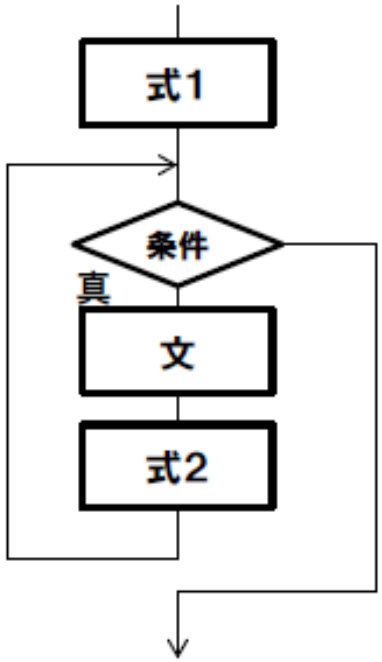
コード	if (条件)
	{
	　成り立つ場合の処理;
	}
	else
	{
	　成り立たない場合の処理;
	}
		switch (式)
		{
		　case 値1:
		　　値1の処理;
		　　break;
		　case 値2:
		　　値2の処理;
		　　break;
		　　・
		　　・
		　　・
		　default:
		　　それ以外の値の処理;
		}
		
		【記述例】
		switch(a) {
		　case0:
		　　xn=100;
		　　break;
		　default:
		　　break;
		　}

処理の流れ	if文に入るとまず条件を調べ、成り立てば「成り立つ場合の処理」を、成り立たなければ「成り立たない場合の処理」を実行してif文を抜ける。	switch文に入るとまず式の値を調べ、その値によって処理を分岐させる。また、caseで指定した値以外の処理はdefault句で指定できる。

for文（指定回数ループ文）

くり返し回数があらかじめ決まっているときに使う

フローチャート	書式

	for (式1;条件;式2)
	{
	　文;
	}

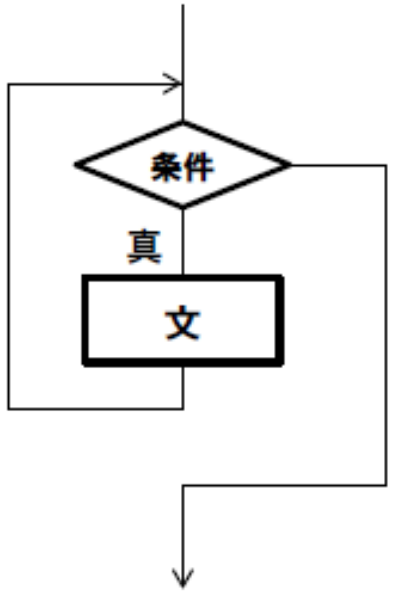
【記述例】

```
for (i=0; i<10; i++) { }   //10回の繰り返し
for ( ; ; ) { }   //無限ループ
```

while文(前判定ループ文)

くり返し回数が決まっていない繰り返しを行う

フローチャート	書式

	while (**条件**)
	{
	　文;
	}

【記述例】

```
while(1) { }  //無限ループ
```

型とその値の範囲

整数型	値の範囲	データサイズ
char（unsigned char型）	-128 ～ 127	1バイト
signed char	-128 ～ 127	1バイト
unsigned char	0 ～ 255	1バイト
short	-32768 ～ 32767	2バイト
unsigned short	0 ～ 65535	2バイト
int	-32768 ～ 32767	2バイト
unsigned int	0 ～ 65535	2バイト
long	-2147483648 ～ 2147483647	4バイト
unsigned long	0 ～ 4294967295	4バイト
long long	-9223372036854775808 ～ 9223372036854775807	8バイト
unsigned long long	0 ～ 18446744073709551615	8バイト

実数型	データサイズ	限界値	
		最大値	正の最小値
float	4バイト	3.4028235677973364e+38f (Ox7F7FFFFF)	7.00649232162408 62e-46f (Ox00000001)
double long double	8バイト	1.7976931348623158e+308 (Ox7FEFFFFFFFFFFFFF)	4.94065645841246 55e-324 (Ox0000000000000001)

残り3つのプログラム

LED PikaTESTは　PikaPikaと同様の動作ですが分岐命令にswitch文を使っています。

```
        while(1){               //無限ループ

        state=0;                //stateを0にリセット
        state |= p4_2;          //SW1の値を取り込む
        state<<=1;              //取り込んだ値を左に1ビットシフト
        state |= p3_7;          //SW2の値を取り込む 00,01,10,11すなわち0,1,2,3になる


        switch(state){          //stateの値により分岐させる
                case 0: p3_4=0,p3_5=0,n=20; break;   //stateの値が0のときの処理
                case 1: p3_4=1,p3_5=0,n=8; break;    //              1
                case 2: p3_4=0,p3_5=1,n=4; break;    //              2
                case 3: p3_4=1,p3_5=1,n=2;           //              3
        }


                LED=0x01;               //変数LEDを0x01にする
                for(i=0;i<8;i++){       //変数iを0から7まで繰り返す
                        p1=LED;         //変数LEDの値をポート1に出力する
                        wait(n);        //変数nでwait関数を実行
                        LED=LED<<1;     //変数LEDの値を1ビット左にシフト
                }
                LED=0x80;               //変数LEDを0x80にする
                for(i=0;i<8;i++){       //変数iを0から7まで繰り返す
                        p1=LED;         //変数LEDの値をポート1に出力する
                        wait(n);        //変数nでwait関数を実行
                        LED=LED>>1;     //変数LEDの値を1ビット右にシフト
                }
                wait(n*2);              //nの2倍時間待つ

        }
}
```

LEDPikaPika_interrupt は高速内蔵クロック(20MHz)とタイマと割り込み処理を使ってより正確な時間を実現しています。

```
////////////////////////////////////////////////////////////////////////
/*  I/Oインプット関数  */                    //入出力ポートの設定をする関数
void init_io(void)
{
        unsigned char i = 0;    // Loop counter

        //* ポートの設定 *//
        p1=0x00;                //ポート1をオール0にする(LED全消灯)
        pd1=0xff;               //ポート1をすべて出力ポートに設定
        p3_4=0,p3_5=0;          //ポート3のビット4とビット5を0にする
        pd3_4=1,pd3_5=1;        //ポート3のビット4とビット5を出力ポートに設定
        pd4_2=0,pd3_7=0;        //ポート4のビット2とポート3のビット7を入力ポートに設定

        //* CPUの動作クロック(20MHz)の設定 *//
        prc0 = 1;               //プロテクト解除
        hocoe = 1;              //高速オンチップオシレータ発振
        for( i=0; i<4; i++){    //発振が安定するまで待つ(最大30us(<125kHzで4クロック分))
                asm("   NOP");      //NOP命令(1クロック分=1/125000秒)を実行して時間を進ませる
        }
        hscksel = 1;            //高速クロックを高速オンチップオシレータにする
        scksel = 1;             //システム基準クロック(fBASE)を高速クロックにする
        prc0 = 0;               //プロテクトをかけておく

        //* タイマRJ2の設定 *//
        msttrj = 0;                             //TRJ2をアクティブ(スタンバイ解除)
        trjmr = 0x10;                   //カウントソース供給(b7=0), f8選択(b6-b4=001),
                                                //片エッジ(b3=0), タイマモード(b2-b0=000)
        trj = 2500-1;                   //1kHz = f8(20MHz/8)/2500
        trjif_trjir = 0;                //TRJ2割り込み要求ビットクリア
        trjie_trjir = 1;                //TRJ2割り込み許可
        ilvlb = 0x01;                   //TRJ2割り込み優先レベル1(b1b0=01)
        tstart_trjcr = 1;               //カウント開始
        asm("   FSET    I");    //割り込み許可
}
```

20MHzクロックを起動

20MHzの1/8=2.5MHzを2500回数えて1msec間隔を作る

タイマ割り込み処理

タイマ:コンペアマッチ機能、インプットキャプチャ機能

```
/************************************************************************/
/*  ウェイト関数  */                    //待ち時間を作るための関数
void wait(unsigned long msec)
{
        flg_1ms = 0;                    //1m秒経過フラグをクリアしておく
        while( msec > 0 ){              //msecが0になるまで繰り返す
                if( flg_1ms == 1 ){     //1m秒経過したら(フラグがセットされたら)
                        flg_1ms = 0;        //1m秒経過フラグをクリアして
                        msec--;             //mescの数を減らす
                }
        }
}

/************************************************************************/
/*  タイマ割り込み関数  */                  //1m秒が経過したことを知らせる関数
#pragma interrupt/B int1ms(vect=22)     //int1msはTRJ2割り込み関数である
void int1ms(void)
{
        trjif_trjir = 0;                    //TRJ2割り込み要求ビットクリア
        flg_1ms = 1;                        //1m秒経過フラグをセットする
}

/************************************************************************/
/*  メイン関数  */                  //メインのプログラム
void main(void)
{
        int     i;                  //iという変数をint型(16bit)で定義
        char    LED;                //LEDという変数をchar型(8bit)で定義
        unsigned long   n;          //nという変数を符号なしlong型で定義

        init_io();                  //init_io()関数を実行

        while(1){                   //無限ループ

    if      (p4_2==0 && p3_7==0) p3_4=0,p3_5=0,n=1000;   //SW1=0,SW2=0ならLED10オフ、LED9オフ、待ち時間nを1000m秒とする
    else if (p4_2==0 && p3_7==1) p3_4=1,p3_5=0,n=500;    //SW1=0,SW2=1ならLED10オン、LED9オフ、待ち時間nを500m秒とする
    else if (p4_2==1 && p3_7==0) p3_4=0,p3_5=1,n=250;    //SW1=1,SW2=0ならLED10オフ、LED9オン、待ち時間nを250m秒とする
    else                         p3_4=1,p3_5=1,n=125;    //上記以外ならLED10オン、LED9オン、待ち時間nを125m秒とする
```

1msec経過フラグがセットされたら、1ひく

タイマ割り込み1msecが入るとフラグをセットする

待ち時間nにnmsecの値を入れる

LEDPikaPika_hotaru は　さらにもう1本タイマを使いPWMのデューティ(明るさ)と点灯時間の2つを制御しています

Lightness: PWMデューティサイクル　1から2500の数値(1が全灯、2500が消灯)
　　100Hzの周期信号

と

点灯時間:msec単位

を変えて蛍らしくしてみましょう。

```
////////////////////////////////////////////////////////////////////////
/*  メイン関数  */              //メインのプログラム
void main(void)
{
        int lightness;              //lightnessという変数をint型(16bit)で定義
        init_io();                  //init_io()関数を実行

        while(1){                   //無限ループ
                lightness = 25000;      //明るさ0%(=1-25000/25000)を設定  ※LEDを消しておく時間を設定する
                trcgrb = lightness;
                trcgrc = lightness;
                trcgrd = lightness;
                wait(1000);             //1000ミリ秒待つ
                lightness = 22500;      //明るさ10%(=1-22500/25000)を設定  ※LEDを消しておく時間を設定する
                trcgrb = lightness;
                trcgrc = lightness;
                trcgrd = lightness;
                wait(800);              //800ミリ秒待つ
                lightness = 20000;      //明るさ20%(=1-20000/25000)を設定  ※LEDを消しておく時間を設定する
                trcgrb = lightness;
                trcgrc = lightness;
                trcgrd = lightness;
                wait(400);              //400ミリ秒待つ
                lightness = 1;          // 明るさ100%(=1-1/25000)を設定  ※LEDを消しておく時間を設定する
                trcgrb = lightness;
                trcgrc = lightness;
                trcgrd = lightness;
                wait(600);              //600ミリ秒待つ
                lightness = 20000;              // 明るさ20%(=1-20000/25000)を設定  ※LEDを消しておく時間を設定する
                trcgrb = lightness;
                trcgrc = lightness;
                trcgrd = lightness;
                wait(400);              //400ミリ秒待つ
                lightness = 22500;              // 明るさ10%(=1-22500/25000)を設定  ※LEDを消しておく時間を設定する
                trcgrb = lightness;
                trcgrc = lightness;
                trcgrd = lightness;
                wait(800);              //800ミリ秒待つ
        }
}
```

参考文献

1. MCR
   http://www.mcr.gr.jp/tech/r8cm12a/main00.html

2. C言語による組み込み制御入門講座　大須賀威彦　　電波新聞社

アンケート

今後、やってみたい電子工作は何ですか？

電子オルゴール、ルーレット、デジタルカウンター、温度計
赤外線リモコン、DCモーター制御、ステッピングモーター制御
アラーム時計、ライントレースカー　、音声出力、電子はかり　など

ルネサス エレクトロニクス株式会社