

IchigoJam^{いちごじゃむ}で プログラミング

センサースイッチ対応版

MISSION CARD POSSIBLE! CARD

君だけの
小っちゃな
コンピュータ。

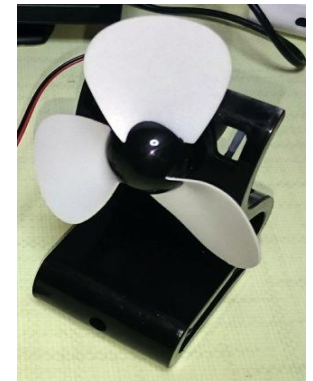


せんぷう機 を便利にまわそう!

100円シOPPのせんぷう機をプログラムで、多機能に変身させます。
モノづくりとプログラミングのはじめの一步!
おもしろい・たのしいからはじめよう!!

- ・このテキストは、保護者等の適切な指導のもとでのご利用を考えて制作しています。ご利用によるすべての事故や損失に関しては、当方は一切の責任を負いません。
- ・本資料はCCライセンスならびに以下の規定にしたがって、複製・改変・再配布することが可能です。著作権は放棄していません。
- ・「IchigoJam」は株式会社 jig.jp の登録商標です。
- ・タイトル、写真などに含まれる「IchigoJam」の称呼は全て株式会社 jig.jp の商品を示しています。
- ・本資料はNPO法人NEXTDAYの協力のもとNPO法人小樽青少年の科学の芽を育てる会が作成しました。
- ・資料の作成にあたり以下の資料を参照しました。
 - >親子でベーシック入門 IchigoJamではじめてのプログラミング (出版社: ジャムハウス)
 - >IchigoJamでプログラミング (発売: プログラミングクラブネットワーク)
- ・原稿についてはICHIGOJAM開発者福野 泰介様のブログを参照させて頂いています。

2023.9.18 Ver.3.05



小樽別院 寺子屋教室
主催 小樽青少年の科学の芽を育てる会
協力 浄土真宗本願寺派 本願寺小樽別院
協力 NPO法人NEXTDAY

IchigoJamを動かそう！

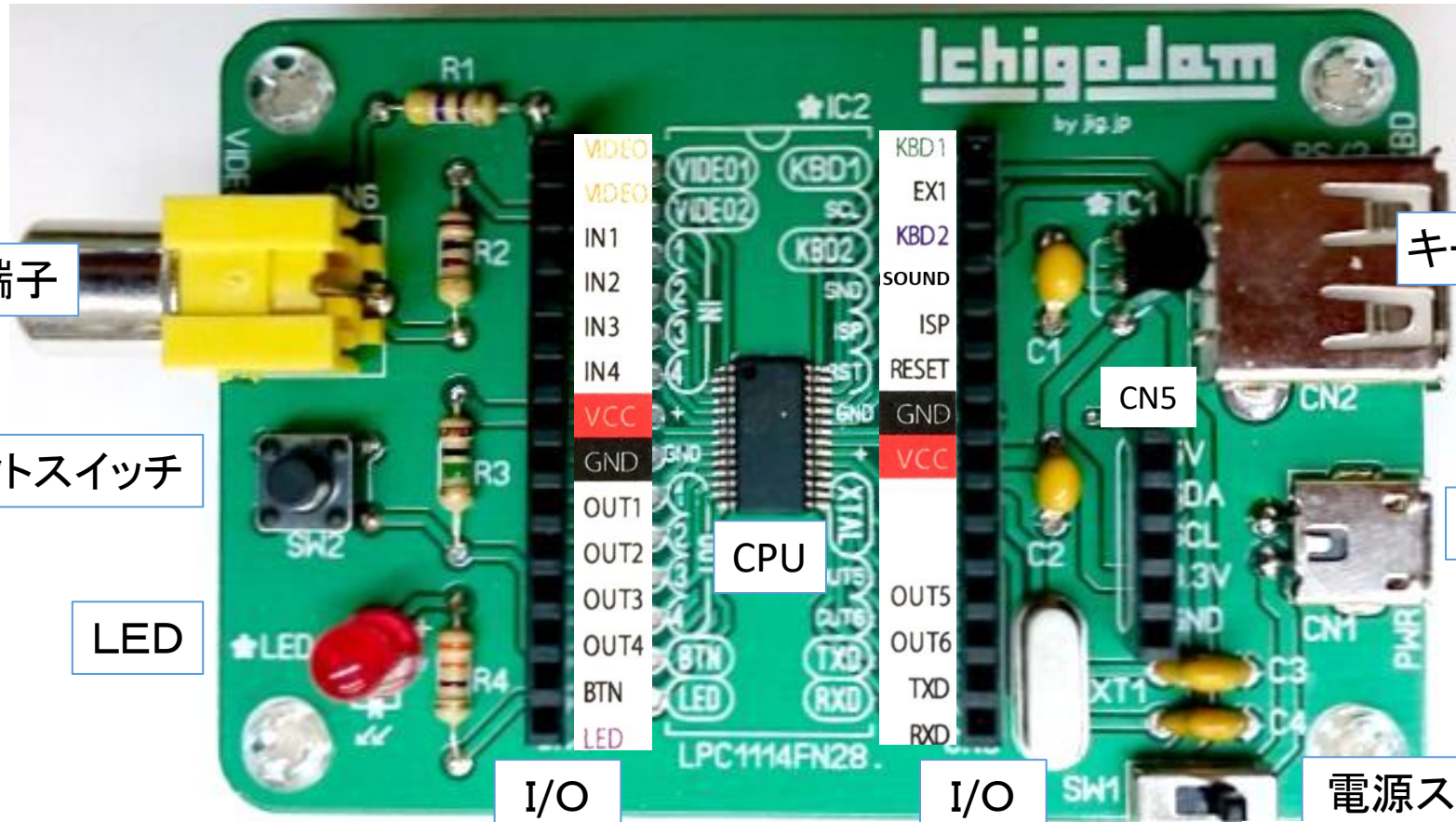
・IchigoJamパソコンの部品配置と名前を覚えよう



液晶モニター



PS/2キーボード



ビデオ端子

タクトスイッチ

LED

I/O
端子

CPU

I/O
端子

CN5

キーボード端子
(変換アダプター)

電源端子
マイクロUSB
5V

電源スイッチ

入 切

◆電源スイッチを入れよう

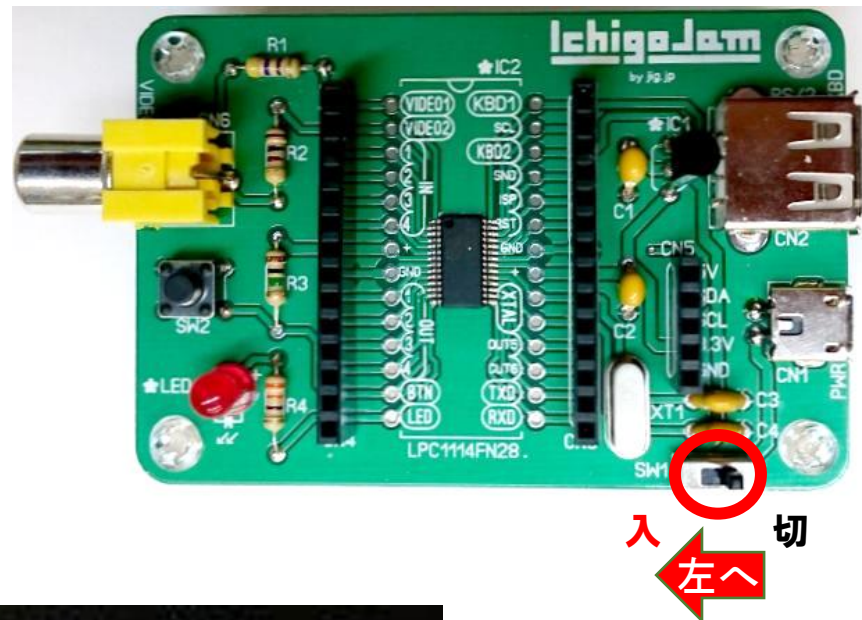
a. 電源スイッチを 右から左 へ動かす



b. 起動すると画面に文字が表示されます



ピッ！ と起動音が鳴ります。



```
IchigoJam BASIC 1.2.1 by jig.jp
OK
```



命令(コマンド)が正しく動作したときに表示されます。

IchigoJamから「わかったよ!」「動いたよ!」と話して(表示して)います。

```
Syntax error
```

シンタックス

エラー

「分からないよ!!」時

きょうのミッション(指令)

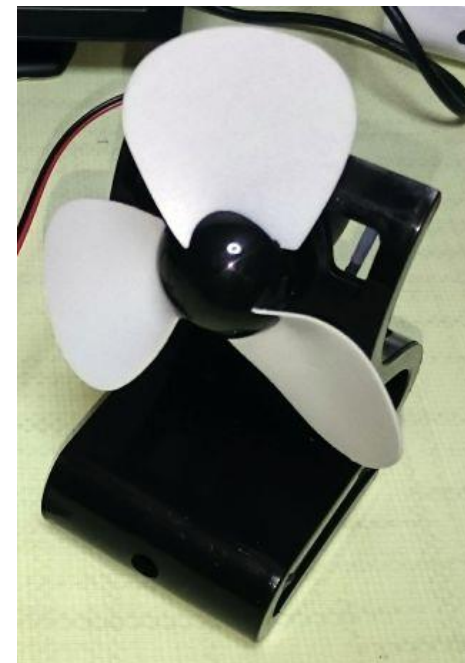
Mission



せんぷう機 を便利にまわそう！

- 1 せんぷう機を動かしてみよう
- 2 どんな機能があると便利かなあ
- 3 動き方やと機能を考えよう
 - ・電源スイッチ
 - ・風量調節
- 4 プログラムで スイッチ/ボタン をつくろう
- 5 多機能せんぷう機に変身！！

80分



IchigoJam





せんぷう機を便利に動かそう！

IchigoJamの電気ではモーターを直接まわすことはできません。
そこでトランジスタの仲間の MOSFET を使かい、
乾電池からの電気の量を調節してモーターをまわします。



パワーMOSFET 2SK4017 を使い
DCモーターに流す電源スイッチの役割りをプログラムします。
このMOSFETは、力のあるモーターを回したり、たくさんのLED点滅な
どの大きな電力を必要とするときに利用され、低電圧でも効率よく動
かすことができます。

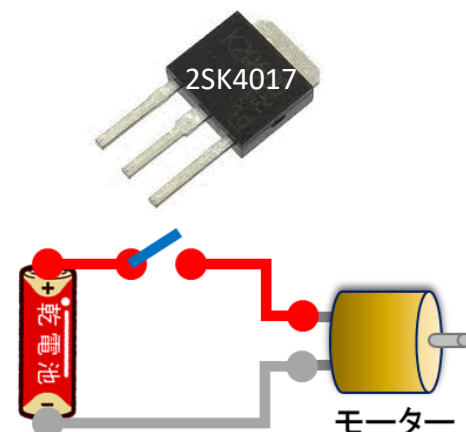
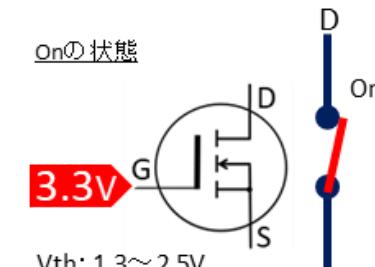
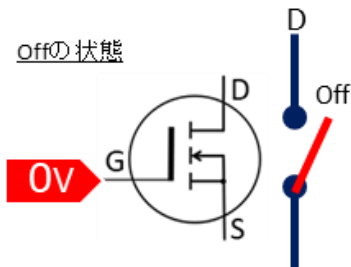
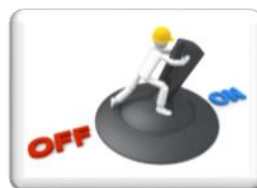
パワーMOSFET

電源スイッチのOn/Offの仕組み

Offの状態 OUT2から出力されない(0V)時、
D-S間のスイッチは切られています。

Onの状態 OUT2から出力された(3.3V)時、
D-S間のスイッチはつながります。

このONとOFFを高速で切り替えると、
流れる電気の量を調節できます。
この切替はプログラムで行ないます。



Step 1 モーターのまわそう！

コンピュータで回転を制御するプログラムをつくりま^{せいぎょ}す

せんぷう機をつなごう



せんぷう機の配線



CN5のGND
一番下の端子

OUT2

コマンドで制御

OUT2, 1

モーターが回転

OUT2, 0

モーターが停止

PWM2, 1000

PWM2, 0



この数値を
500～2000までの範囲で
変えて見よう！

Step 2 せんぷう機にはどんな機能があるかな？

身近なせんぷう機を見て確認します。

- ・電源、運転
- ・風量(弱/中/強)
- ・リズム
- ・タイマー
- ・首ふり



こうした機能をプログラムにする場合、動作や動き方を細かく分解して考えます。

■電源の場合

- ・ボタンを押さない
 - ・ボタンを押す
- 監視 ⇒ ライトを消す ⇒ 電気を切る
視断 ⇒ ライトが点く ⇒ 電気を流す



■風量(弱/中/強)の場合

- ・弱 ⇒ モーターがゆっくりまわる ⇒ 電気の流れる量が少ない
- ・強 ⇒ モーターが早くまわる ⇒ 電気の流れる量が多い

PWM2, 800

PWM2, 1800



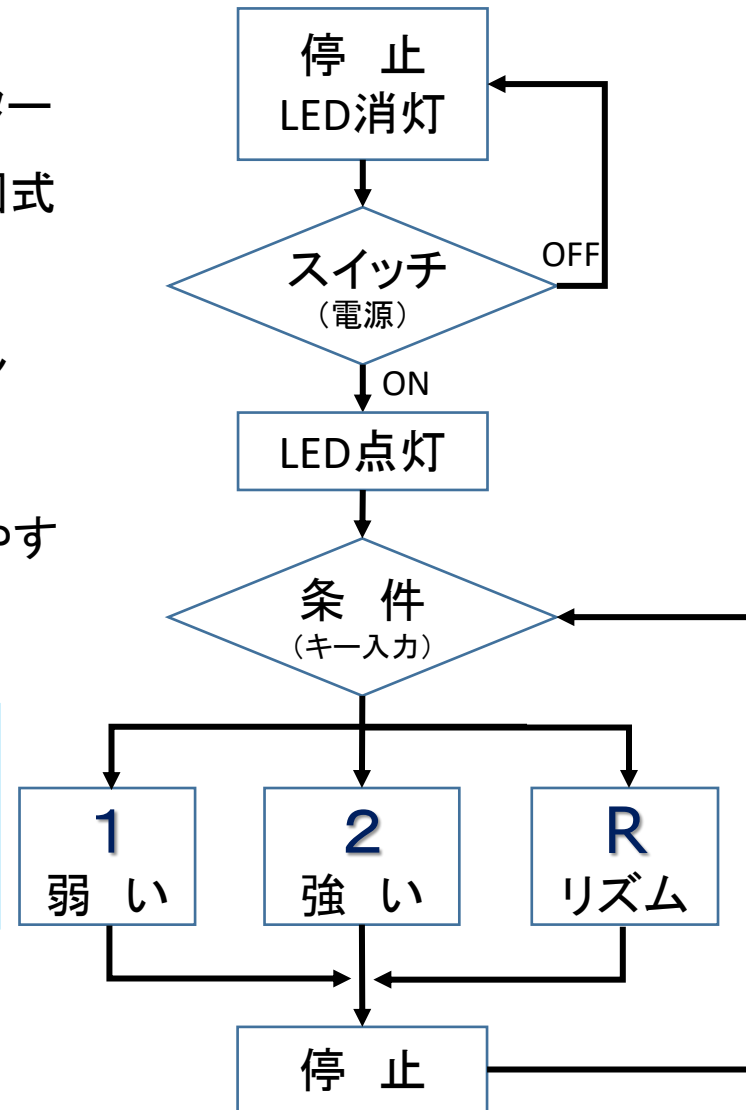
Step 3 せんぷう機の機能と動き方を決めよう！

プログラムをつくるときは、
「何を」「どの順番で」「どのように動くのか」をコンピューター
に分かるように書く必要があります。それらを分かりやすく図式
にしたのを アルゴリズム といいます。
細かな動作や働きを分解することで、どのような命令(コマン
ド)や機器が必要かがわかります。
思いどおりに動かない時やコマンドの間違いなどを見つけやす
くなります。

今回は、

- ・電源ボタン
 - ・風量 (弱/強/リズム)
- の2つの機能を
プログラムで動かすことに挑戦します。

アルゴリズム (手順)



電源ボタン 機能をつくる

Step 4 センサーを付けよう

センサースイッチ を端子に接続します。
センサースイッチがどのように反応するか、
次のコマンドでコンピューターの想いを見よう！



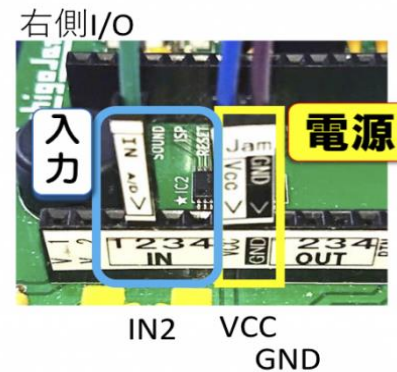
センサースイッチを

- ・ 押しながら
- ・ 押さないで

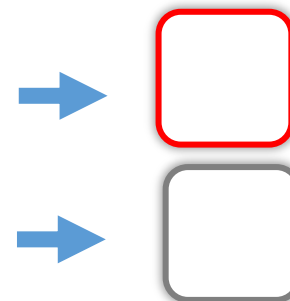
センサースイッチ

Step 5 プログラムでまわそう

```
25 LED 1
40 PWM2,1000:WAIT60
50 PWM2,0:WAIT30
60 LED 0
```

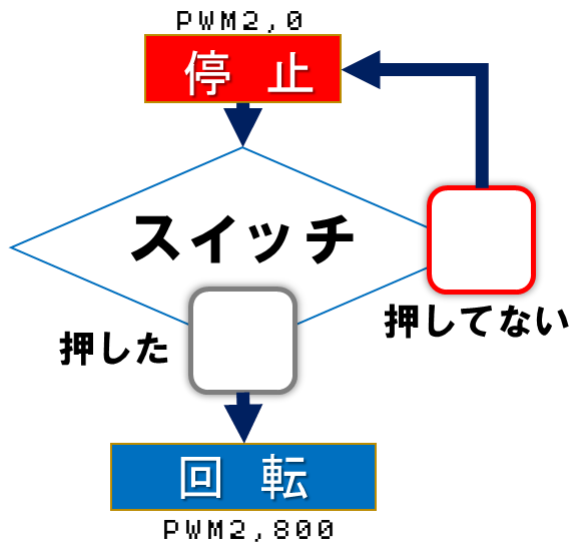


モニターの数値



```
LOAD 1
LIST
```





◆センサースイッチの数値(0か1)をコンピューターに判断させてモーターを回すようにします。

もし ●●● ならば □□□ する ●●●には条件を書きます
 I F ●●● THEN □□□ □□□には動作を書きます

もし ならば
 I F IN(2)=0 THEN PWM2,0
 スイッチが 停止する
 押していなければ

20行 と 90行 にプログラムを追加しよう！

電源スイッチ

```

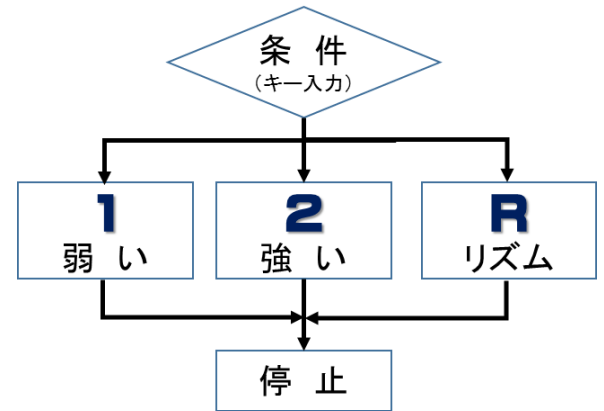
追加 → 20 IF IN(2)=0 THEN PWM2,0:GOTO20
      25 LED 1
      40 PWM2,1000:WAIT60
      50 PWM2,0:WAIT30
      60 LED 0
追加 → 90 GOTO 20
  
```

プログラムを
 修正したら エンター
 キー

プログラムを
 追加・修正・さくじょ した時は
 必ず LIST コマンドで確認！

・風量（弱/強/リズム） 機能をつくる

キーボードの一つ一つに 数値 が割りあてられています。
その数値を キーコード といいます。
そのキーコードに 風量(弱/強/リズム) とを関連付けることで
選択ボタンとして機能させます。

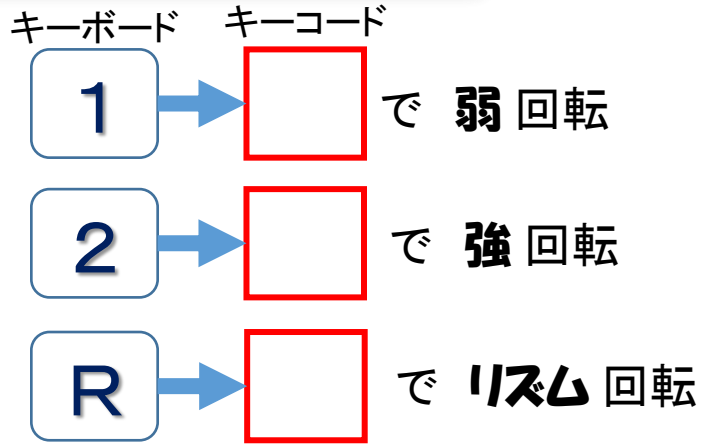


30 A=INKEY() : ? A

```
20 IF IN(2)=0 THEN PWM2,0:GOTO20  
25 LED 1  
40 PWM2,1000:WAIT60  
50 PWM2,0:WAIT30  
60 LED 0  
90 GOTO 20
```

Step 6 キーコードで風量を選べるようにしよう！

プログラムを実行して、
・数字の 1 と 2
・英字の R
のキーコードを調べよう。



Step 7 プログラムを完成させよう!

プログラムを
修正したら エンター
キー

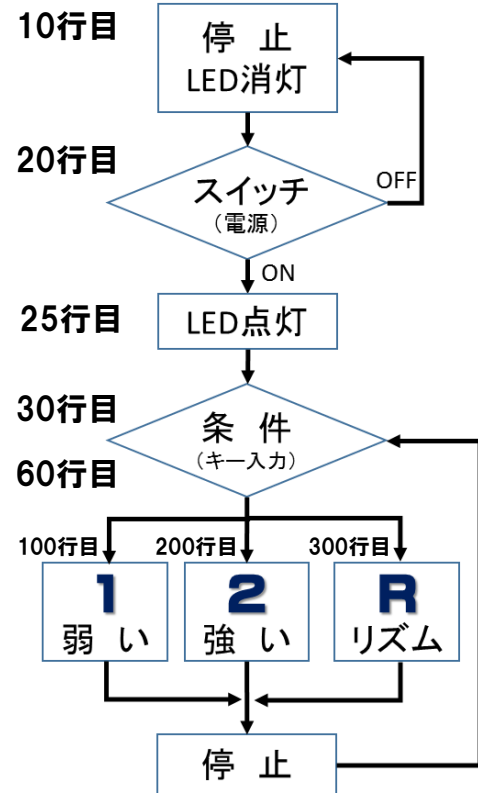
プログラムを
追加・修正・삭じよした時は
必ず LIST コマンドで確認!

LOAD 3
LIST



3番のプログラムを読み込みます

キーコードを使いどのキーが押されたかを
プログラムで判断させて動かします。



```

10 PWM2,0:LED0
20 IF IN(2)=0 THEN PWM2,0:GOTO20
25 LED1:WAIT2:BEEP 30,60
30 A=INKEY():?A
40 IF A= [ ] THEN GOSUB 100
50 IF A= [ ] THEN GOSUB 200
60 IF A= [ ] THEN GOSUB 300
90 GOTO 30
  
```

← プログラムが動いた時の、一番初めの状態

← 電源スイッチがONになるとLED点灯

← キーボードからのキーコードを取得

キーコードによって実行する
プログラムに移動

```

100 PWM2,800:WAIT300
110 PWM2,0:BEEP:RETURN
  
```

1
弱い

```

200 PWM2,1800:WAIT300
210 PWM2,0:BEEP:RETURN
  
```

2
強い

```

300 FOR I=1 TO 10 ← 10回繰り返す
310 B=RND(10) ← 0~9までの乱数がでる
320 PWM2,500+B*80:BEEP:WAIT30
330 NEXT ← 乱数によって電気の量が自動的に変化する計算式
340 PWM2,0:BEEP:RETURN
  
```

R
リズム

Step8 プログラムをどんどん改造しよう！

※終了処理を考えて追加してみましょう。

スイッチをもう一度押したらLEDを消してプログラムを終了します。

※いろいろ音色を変えてみましょう。

BEEPコマンドの値を変更する！

※それぞれの速度を変えてみましょう。

PWMコマンドの値を変更する。

※それぞれの時間を変えてみましょう。

WAITコマンドの値を変更する。



自分のオリジナル改造を
してみましょう。

解らない時は講師の先生
に聞いてみましょう。

さまざまなプログラムに挑戦！！

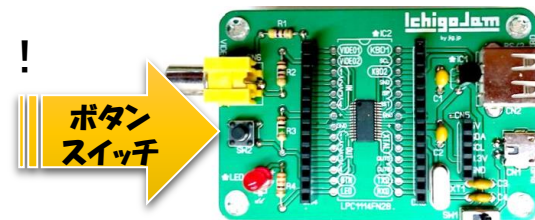
```
10 ?BTN()  
20 LED BTN()  
30 GOTO 10
```

```
10 CLS  
20 FOR I=1 TO 10  
30 A=I+1: ?A  
30 GOTO 20
```

```
10 N=3:C=60:CLT:CLS:LED 0  
20 T=N*60*C-TICK()  
30 IF T<=0 LED 1:END  
40 S=T/C:M=S/60:S=S%60  
50 LC 0,0: ? M;"m";S;"s ":GOTO 20
```

※新しいプログラムを入力するときは、
コマン **NEW** で
コンピューターの記憶を消してから！

IchigoJamのボタンスイッチを押すと！



TO 10 を
TO 100 や TO 1000 に変更してみよう！

インスタントラーメンタイマー

どんなメロディが聞こえるかなあ？

```
10 PLAY"O4L8 E2E4D4E4ED>B2<F4FFA4FEFEDDE2"
```

コマンドをおぼえよう !!

プログラムをリストするには

LIST ↵

範囲を指定してリスト
LIST 10,100

プログラムをすべて削除するには

NEW ↵

プログラムを実行するには

RUN ↵

画面表示をすべて消します (クリア スクリーン)

CLS ↵

プログラムをメモリーに記憶させるには

SAVE ↵

(0~3)の4個のメモリーがあります

プログラムの行削除は行番号だけを打ちます

10 ↵

プログラムをメモリーから読みだすには

LOAD ↵

メモリーの内容は電源を切っても記憶しています

実行しているプログラムを停止するには



キーを押します

エスケープと読みます
(Escape: 逃れる、抜け出す、
脱出する)の省略型です

メモリーの内容をみるには

FILES ↵

コマンドはキー入力だけでなくファンクションキーでも代用できるよ



| | | | | | | | | |
|-----|------|------|------|-----|---------|------|--------|-------|
| F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 |
| CLS | LOAD | SAVE | LIST | RUN | ?FREE() | OUT0 | VIDEO1 | FILES |

IchigoJam BASIC 1.1 cheatsheet

*赤字は省略可能を示す *青文字は説明文

ファンクションキー

| | | | | | | | | |
|-----|------|------|------|-----|---------|------|--------|-------|
| F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 |
| CLS | LOAD | SAVE | LIST | RUN | ?FREE() | OUT0 | VIDEO1 | FILES |

| Dec | Hex | Bin |
|-----|-----|------|
| 0 | #00 | 0000 |
| 1 | #01 | 0001 |
| 2 | #02 | 0010 |
| 3 | #03 | 0011 |
| 4 | #04 | 0100 |
| 5 | #05 | 0101 |
| 6 | #06 | 0110 |
| 7 | #07 | 0111 |
| 8 | #08 | 1000 |
| 9 | #09 | 1001 |
| 10 | #0A | 1010 |
| 11 | #0B | 1011 |
| 12 | #0C | 1100 |
| 13 | #0D | 1101 |
| 14 | #0E | 1110 |
| 15 | #0F | 1111 |

4bit : 0~15
8bit : 0~255
(-128~127)
16bit : 0~65535
(-32768~32767)

*数える数値は16ビット範囲
小数値は見えません。

| | |
|--------|-------|
| VIDEO1 | KBD1 |
| VIDEO2 | EX1 |
| IN1 | KBD2 |
| IN2 | SOUND |
| IN3 | ISP |
| IN4 | RESET |
| VCC | GND |
| GND | VCC |
| OUT1 | - |
| OUT2 | - |
| OUT3 | OUT5 |
| OUT4 | OUT6 |
| BTN | TXD |
| LED | RXD |

演算の優先順位
高い () ~-! NOT * / % MOD <<>> & ^ + - = != <> <=> = AND OR 低い

●式/演算

[加算] 数+数
[減算] 数-数
[乗算] 数*数
[除算] 数/数
[剰余] 数%数
[否定] NOT式
[論理積] 数&数
[論理和] 数|数
[排他的論理和] 数^数
[右シフト] 数>>数
[左シフト] 数<<数
[ビット反転] ~数
[優先順位変更] (~)
式AND式 省略形:&&
式OR式 省略形:||

●関数

ABS(数)
ASC("文字")
BIN\$(数,桁数)
CHR\$(数,...数n)
DECS(数,桁数)
HEXS(数,桁数)
RND(数)

●数値表記

123 10進数
(-32768~32767)
#E9 16進数(0~#FFF)
^1001 2進数

●定数

LEFT 左 : 28
RIGHT 右 : 29
UP 上 : 30
DOWN 下 : 31
SPACE 空白 : 32

●条件判断/条件式

IF 数 THEN 次 ELSE 次 2
【等しい】 数1=>数2
【等しくない】 数1<>数2
【小さい】 数1<数2
【小さいか等しい】 数1<=数2
【大きい】 数1>数2
【大きいか等しい】 数1>=数2

●移動/繰り返し/サブルーチン

FOR 変数=数1 TO 数2 STEP 数3~NEXT
GOSUB 行番号 省略形:GSB
GOTO 行番号
RETURN 省略形:RTN

●ハードウェア

ANA(数) 0~1023
BPS 通信速度 省略時:115,200bps
I2CR(数1,数2,数3,数4,数5)
I2CW(数1,数2,数3,数4,数5)
IN(数) IN1-9から入力
LED 数 0:消灯,1:点灯
OUT 数1,数2 OUT1-7に出力
PWM 数1,数2,数3
RESET リセット
SLEEP スリープ(ボタンを押すと復帰)
UARTシリアル出力設定,シリアル受信設定
WAIT 数 60で約1秒

●ファイル

FILE()
FILES 数1,数2
LOAD 数
LRUN 数,行番号
RUN
SAVE 数

●プログラム

CONT 再度実行する
END プログラムを終了
FREE() プログラムの残りメモリ数
LINE() 現在実行中の行番号
LIST 行番号1,行番号2
NEW プログラムを消す
RENUM 数1,数2
STOP 処理を中断する

●メモリ操作/マシン語

PEEK(アドレス)
POKE アドレス,数,...数n
USR(アドレス,数)

●その他

HELP メモリマップを表示
REM 注釈 省略形:'
TICK() tick時間(1/60)を返す
VER() バージョン番号を返す

#0000 メモリマップ
#0700 文字パターン(#00~#DF)
#0800 PCGパターン(#E0~#FF)
#0900 配列変数・変数
#0C00 画面(32文字×24行)
#1001 プログラムリスト
#1002 キーが押されたビット
#1003 キーバッファ格納数(最大14)
#1004~F キーバッファ

●音楽/サウンド

BEEP 高周,長さ BEEPを鳴らす
PLAY [MML] MMLなしで演奏停止
SOUND() 再生中なら1を返す
TEMPO テンポ テンポを指定

■MML (Music Macro Language)

[音] 音(CDEFGABR)
[音]n 音長(.を付けると1.5倍長)
[音]+ 半音上げる
[音]- 半音下げる
Tn テンポ(初期値:120)
Ln デフォルトの音長(初期値:4)
On オクターブ指定(1~5)
> 1オクターブ上げる
< 1オクターブ下げる
\$ 以後のMMLを繰り返す
Nn 音の高さを指定
(音長で指定可能な値:1,2,3,4,8,16,32)

●制御(コントロール)コード

08(#08) バックスペース(後退)
13(#0D) リターン
14(#0E) インサート(挿入)
127(#7F) デリート(削除)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|----|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 1 | ! | @ | # | \$ | % | & | ' | (|) | * | + | , | - | . | / | : |
| 2 | ; | < | = | > | ? | [| \ |] | ^ | _ | ` | { | | } | ~ | ? |
| 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 4 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 5 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 6 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 8 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| A | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| B | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| C | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| D | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| E | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| F | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |

1文字は8×8ドットで構成

「IchigoJam 1.2 チートシート by OpenSpace」

<http://www.openspc2.org/reibun/IchigoJam/etc/cheat-sheet/0002/>