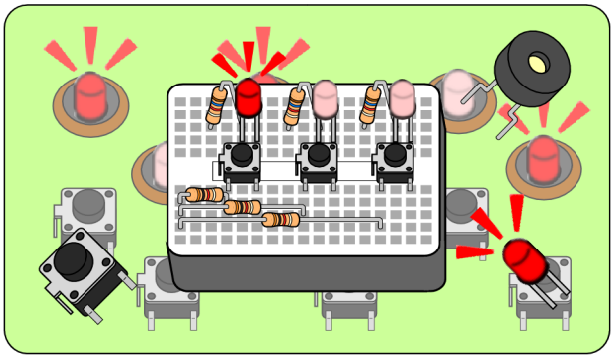
IchigoJam 電子工作パーツセット
LEDもぐらたたき

MOL-jamP

Whack-a-Mole Parts Set

IchigoJam 0.9.7 / 1.0.1対応

概要



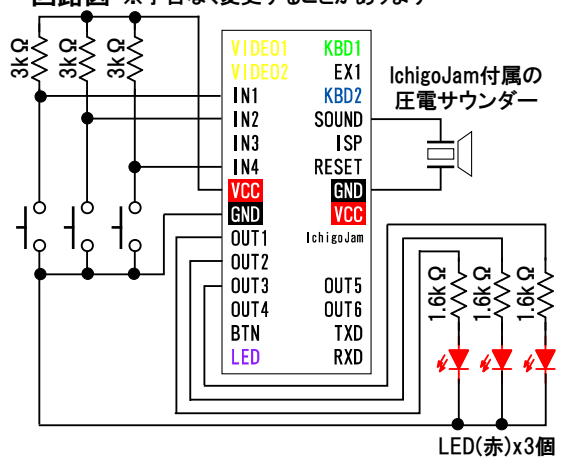
みなさんおなじみのもぐらたたきゲームです。

ランダムに出現するLEDもぐらを、ブレッドボード上のタクトスイッチを押してやっつけましょう。

楽しく遊びながらスイッチの入力とLEDの出力について学習できます。

本キットのプログラムはIchigoJam バージョン0.9.7及びバージョン1.0.1で動作確認しています。

回路図 ※予告なく変更することがあります



部品表 ※予告なく変更することがあります

	品名/型番/値	数量	備考		品名/型番/値	数量	備考
1	LED 赤	3		4	抵抗 3kΩ	3	橙黒赤金の色帯
2	タクトスイッチ	3		5	ジャンプワイヤ 10cm	10	
3	抵抗 1.6kΩ	3	茶青赤金の色帯	6	ブレッドボード	1	

プログラムのセーブとロードのしかた

作成したプログラムをSAVEコマンドで保存することができます。

```
      ⋮
130 GOTO 30

SAVE0 ↵
```

プログラムをSAVEしないと、IchigoJamの電源をOFFにしたときに作成したプログラムが消えてしまいます。

上の例のようにSAVEコマンドの後ろにプログラム番号を指定して実行すると、指定した番号に保存されます。

プログラム番号はIchigoJam本体に保存するとき0～3(バージョン0.9.7では0～2)、外部のEEPROMカセットに保存するときは100～131番になります。

! SAVEすると前に入っていたプログラムは上書きされます(消えてなくなります)ので十分注意してください。

```
LOAD0 ↵
Loaded xxxbyte
OK
```

SAVEコマンドで保存したプログラムは、LOADコマンドでファイル番号を指定すると呼び出すことができます。

☆プログラム番号を省略したとき

プログラム番号を省略してプログラムをロード/セーブすることもできます。このときどの番号のプログラムが対象になるかはIchigoJamのバージョンによって次のように異なります。注意してください。

◎ IchigoJam バージョン1.0.1の場合

SAVEコマンドのときは最後に実行したLOADまたはSAVEコマンドで指定した番号に上書き保存されます。

LOADコマンドのときは最後に実行したLOADまたはSAVEコマンドで指定した番号からプログラムを呼び出します。

(※電源をONしてはじめてLOADコマンドやSAVEコマンドを実行した場合、番号を省略すると0番が対象になります)

◎ IchigoJam バージョン0.9.7の場合

SAVEコマンドのときは0番に上書き保存されます。

LOADコマンドのときは0番からプログラムを呼び出します。

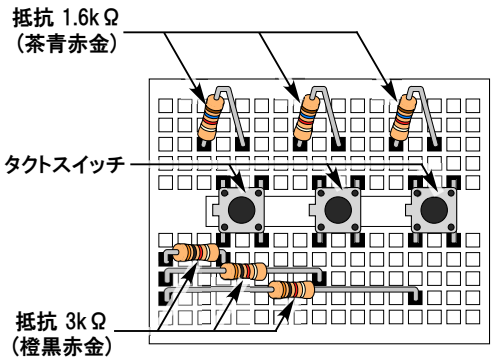
EEPROMカセットを使用する場合

EEPROMカセットからプログラムをロードし、その後EEPROMカセットを取り外した場合、番号を省略してセーブしようとするとファイルエラーになりセーブできません。

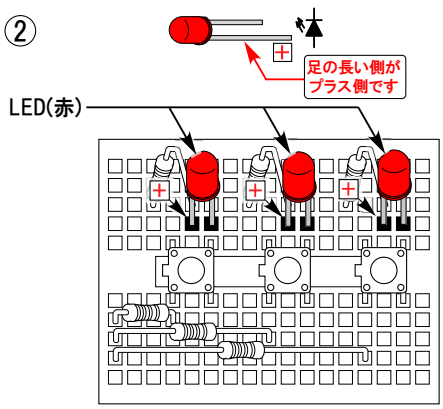
EEPROMカセットを取り外した後の初回のセーブはIchigoJam本体のプログラム番号を指定してIchigoJam本体にセーブしてください。

1 組み立てかた

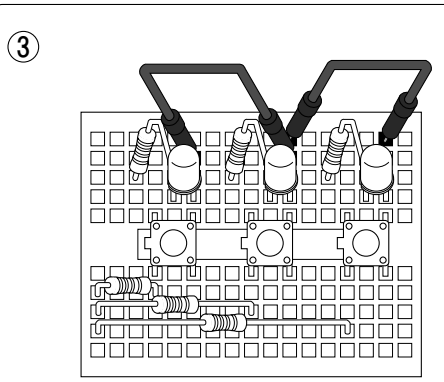
①



②

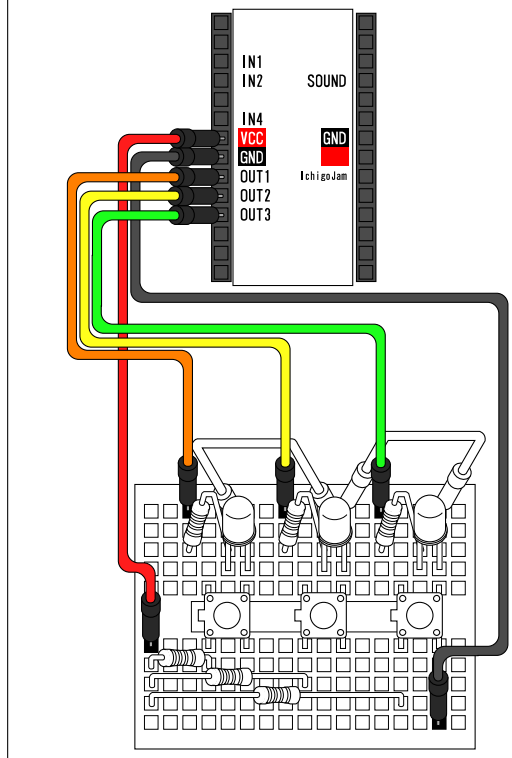


③

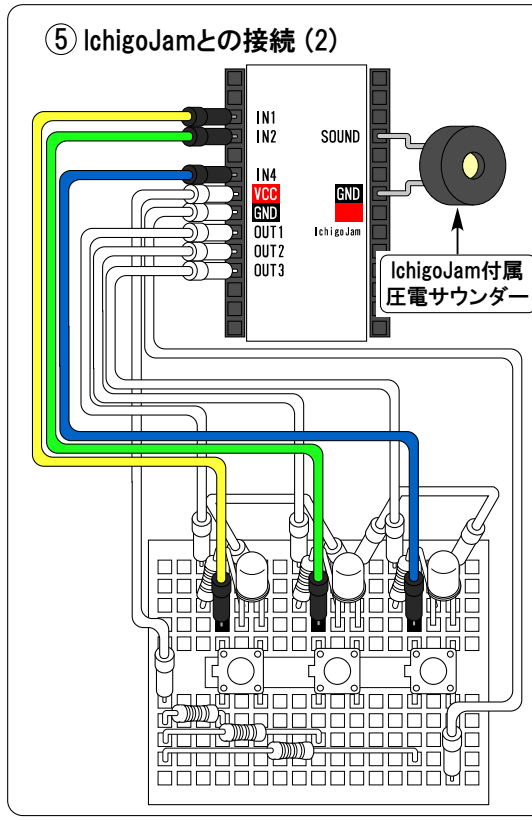


ページ右上④に続きます

④ IchigoJamとの接続 (1)



⑤ IchigoJamとの接続 (2)



2 回路の動作をチェックしよう

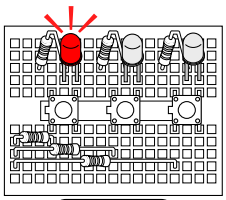
組み立てた回路のチェックを兼ねて、早速動かしてみましょう。

IchigoJamにはBASICのコマンドを即時実行する機能があります。この機能を使用すると、組み立てた回路の動作を手軽に試せます。

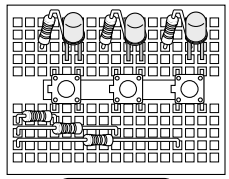
```
OUT 1,1↵
OK
```

行番号なしでBASICの文(コマンド)を入力し、「ENTER ↵」キーを押すと、入力した文がすぐに実行されます。

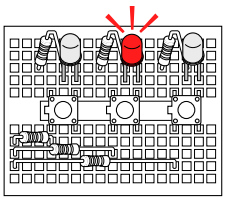
下図のようにコマンドを実行し、図のようにLEDが点灯/消灯することを確認してください。



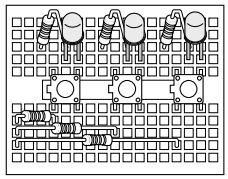
「OUT 1,1」を実行したとき



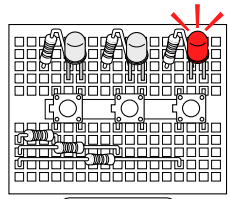
「OUT 1,0」を実行したとき



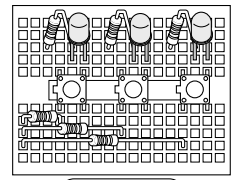
「OUT 2,1」を実行したとき



「OUT 2,0」を実行したとき



「OUT 3,1」を実行したとき



「OUT 3,0」を実行したとき

LEDの点滅操作がうまくいかないときは、ブレッドボード上のLEDまわりの組み立てが間違っていないか確認してください。

次に、入力機能のチェックをします。

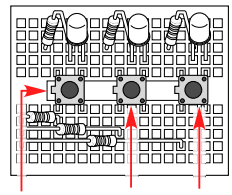
```
PRINT IN(1)↵
1
OK
```

左のように、「PRINT IN(1)」と入力して「ENTER ↵」キーを押すと、IchigoJamのIN1端子につないだスイッチの状態が、モニタ画面に表示されます。

IN1のスイッチを押した状態でコマンドを入力すると「0」が、スイッチを放した状態でコマンドを入力すると「1」が表示されるのが正常です。

同様にIN2のスイッチ、IN4のスイッチについてもチェックしてください。

モニタ画面に表示される数字が「0」のまま(または「1」のまま)変わらないときは、スイッチまわりの組み立てをチェックしてください。



IN1のスイッチ　IN2のスイッチ　IN4のスイッチ

キーワード

- IN()関数
- PRINT文の使い方
- OUT文

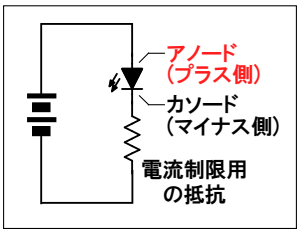
回路と部品の説明

(1) LEDの使い方

LEDは「発光ダイオード」(Light Emitting Diode)の略称で、電子工作の定番部品としておなじみです。使ったことがある人も多いのではないでしょうか?

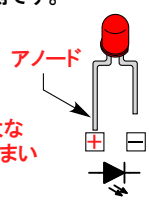
その名の通り、電流を流すと光るダイオードで、トランジスタやICと同じ半導体の仲間です。

発光色は赤、橙、黄、緑、青、白など色々で、中には「ピンク色」や「紫色」といった変り種もあります。



LEDは一方向にしか電流を流しませんので、アノードが回路のプラス側になるように接続します。

キットに入っているLEDを見ると足の長さに長い短いの区別があります。足の長い側がアノード(プラス)側です。



LEDを使用するときは、LEDに流れる電流を制限するための抵抗を直列に入れます。

電流制限用の抵抗を入れないとLEDに過大な電流が流れます。最悪の場合LEDを壊してしまいます。注意しましょう。

直列抵抗の値は、次のように求めることができます。

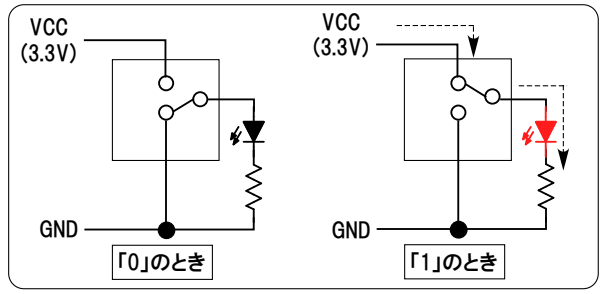
$$\text{直列抵抗の値}(\Omega) = \frac{\text{電源電圧} - \text{LED順方向電圧}(V)}{\text{LEDに流す電流}(A)}$$

LEDの順方向電圧はLEDの色により異なり、赤、黄、黄緑のLEDなら約1.8V～2V程度、純緑、青、白のLEDの場合は約3Vから3.6V程度です。LEDに流す電流は数mA程度が普通です。

(2) IchigoJamの出力ポートのはたらき

IchigoJamにはOUT1～OUT6の6本の出力ポートがあります。出力ポートはBASICのOUT文で操作できます。

出力ポートのはたらきは下の図のように、VCC(3.3V)とGND(0V)の間で切り替わる切り替えスイッチを想像するとイメージがつかみやすいと思います。
(実際にも半導体素子による切り替えスイッチになっています)



OUT文で出力ポートを「0」にすると、IchigoJam内部の切り替えスイッチは「GND(0V)」側に切り替わります。本キットのように出力ポートとGND間にLEDを接続してある場合、LEDには電流は流れません(消灯しています)

出力ポートを「1」にすると、内部の切り替えスイッチは「VCC(3.3V)」側に切り替わります。するとLEDに電流が流れ、LEDが点灯します。

キーワードの説明

OUT文

OUT文は、IchigoJamの出力端子(OUT1～OUT6)を操作する命令です。「OUT」に続く最初の数字(1～6)で操作する出力端子(OUT1～OUT6)を選び、2番目の数字(0または1)で出力端子の状態を指定します。

(例) OUT 1,1

上の使用例では、OUT1端子に「1」を出力します。「1」を出力すると、OUT端子が約3.3V(「H」レベル)になります。「0」を出力した場合はOUT端子が約0V(「L」レベル)になります。

OUT文を実行したとき、OUT文で選んでいないOUT端子の状態は変化しません。

参考（複数のOUT端子を同時に設定する方法）

上の説明では個々のOUT端子を別々に設定していますが、複数のOUT端子をまとめて同時に設定することもできます。

「OUT」の後に「0～63」の数字を1つだけ書くと、IchigoJam内部で2進数に変換され、OUT1～OUT6端子に同時に出力されます。

(例) OUT 15

この例ではOUT1～OUT4端子を「1」に、OUT5、OUT6端子を「0」に設定します。

OUT1～OUT6端子を全部「0」に設定するには、次のように書きます。

(例) OUT 0

IchigoJam バージョン1.0.1では数の表現として、10進表記以外に16進表記と2進表記も使用できます。用途に合わせて使用するとプログラムがわかりやすくなり便利です。

数字の頭に何も指定しない場合は10進表記です。
数字の頭に「#」記号を指定した場合は16進表記、「`」(バッククオート)記号を指定した場合は2進表記です。

16進表記は「0～9の数字」と「A～Fのアルファベット」で数を表現します。

2進表記は「0と1の数字」の組み合わせで数を表現します。

PRINT文の使い方

PRINT文は変数の値(計算結果)やメッセージを表示するのに使用します。
基本的な使用方法は次のとおりです。

(例) PRINT K

この例では変数Kの値を画面に表示します。

(例) PRINT ″ﾎﾝｼﾞﾂﾊ ｾｲﾃﾝﾅﾘ″

この例では「ﾎﾝｼﾞﾂﾊ ｾｲﾃﾝﾅﾘ」というメッセージの文字列を画面に表示します。文字列はダブルクオーテーション「″」で囲みます。

PRINT文で表示させる項目をセミコロン「;」記号で区切って続けて書くと、複数の項目を1行に表示させることができます。

(例) PRINT W/10;″.″;W%10;″V″

この例ではまず変数Wの値を10で割った値(小数点以下は切捨てとなります)を表示し、そのあとに小数点記号(ピリオド「.」記号)を表示し、そのあとに変数Wの値を10で割った余りを表示、最後に文字「V」を表示します。

PRINT文の後ろに何も指定しない場合は、何も表示せず、改行のみ行います。

参考（プログラム中のカタカナについて）

IchigoJamにPCを接続し、PC上の端末ソフトからプログラムを入力して使用している人も多いと思います。

この場合、端末ソフトの設定によってはREM文やPRINT文中で使用しているカタカナが文字化けして正常に表示されなかったり、「Syntax Error」になってうまく実行できなかったりすることがあります。

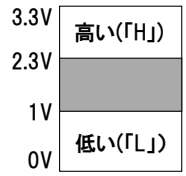
これはIchigoJamのカタカナの文字コードとPC側の文字コードが一致しないため、端末ソフト側で別の文字になったり「不明の文字」として「?」に置き換えられたりするために起こります。

端末ソフトの設定で文字コードが「UTF-8」になっている場合は「JIS」または「Shift-JIS」に変更すると比較的うまくいくようです。それでも直らない場合はカタカナの部分をローマ字表記にするなどして対策してください。

※本キットのプログラムはIchigoJamに接続したキーボードから直接入力する前提で作成しています。

IN()関数

IN()関数は、指定した入力端子(IN1～IN4)の電圧が高い(「H」レベル)か低い(「L」レベル)かを調べ、値を返す関数です。



電圧が高い(「H」レベル)のときは「1」を、電圧が低い(「L」レベル)のときは「0」を返します。

「H」レベル、「L」レベルの入力電圧範囲については左の図を見てください。図の灰色の部分はレベルが「H」なのか「L」なのか特に決まっていない部分なので、使用しません。

(例) PRINT IN(1)

上の例は、IN1端子の電圧の状態をPRINT文でモニタ画面に表示します。IN()関数のかっこ「()」内に電圧の状態を調べたい入力端子の番号を指定します。

3 LEDもぐらをランダムに出現させよう

実際にもぐらたたきゲームを作る下準備として、LEDもぐらをランダムに出現させます。

LEDの点滅操作を自動化するためには、プログラムを書きます。

プログラムリスト

```
10 PRINT "MOGURA TATAKI"
20 OUT 0:D=30:N=25
30 OUT 0:X=RND(7)+1:WAIT D
40 IF (X%2)=0 OUT 1,0:GOTO 60
50 BEEP 10,5:OUT 1,1:N=N-1
60 IF ((X/2)%2)=0 OUT 2,0:GOTO 80
70 BEEP 12,5:OUT 2,1:N=N-1
80 IF ((X/4)%2)=0 OUT 3,0:GOTO 100
90 BEEP 15,5:OUT 3,1:N=N-1
100 WAIT D
110 IF N>0 GOTO 30
120 IF BTN()=0 THEN GOTO 120
130 GOTO 20
```

キーワード

- 変数 D
- IF文と条件判断
- GOTO文
- WAIT文
- RND()関数
- 割り算の余り
- BEEP文
- BTN()関数

プログラムの説明

(1) プログラムとは何か?

最初の実験のときはLEDを点滅させるためにために手動でBASICの命令を入力しました。このBASICの命令を「プログラム」にすることで、LEDを点滅させる操作を自動化できます。

「プログラム」とは、コンピュータ(IchigoJamも立派なコンピュータです)の命令をある決まりごと(文法)に従って書いたものです。コンピュータはプログラムに書いてある「ああしてこうしろ」という命令を、「プログラムに書いてある通り」に実行します。

コンピュータの命令と書き方の決まりごとのことをまとめて「プログラミング言語」といいます。プログラミング言語にはいろいろありますが、IchigoJamの場合は、「BASIC」というプログラミング言語でプログラムを書きます。

(例) 200 GOTO 20 「プログラムの20行にジャンプせよ」という文です。
(200: 行番号, GOTO: 命令)

BASICでは、まず行番号を書き、行番号に続いて命令(文)を書くという決まりがあります。

行番号をつけてBASICの命令を入力すると、入力した命令はすぐには実行されず、IchigoJam内部のプログラム用のメモリに置かれます。

プログラムを書いたあと、行番号なしで「RUN」命令を入力すると、行番号の若い順にプログラムが順次実行されます。プログラムが決まりごとに違反している場合は、「Syntax Error」などのエラーメッセージを出して止まります。

---

(2) プログラムの動作について

もぐらたたきゲームで最も重要な、「もぐらをランダムに出す」という部分をプログラム化します。

① モニタ画面にタイトルを表示したあと、LEDを全消灯させ、変数に初期値を入れます。
変数Dは次のLEDもぐらを出すまでの待ち時間(0.5秒)、変数NはLEDもぐらの総数(25匹)です。

② 1から7までのランダムな数(X)を求め、次の条件でLEDもぐらを出します。

LEDもぐら1の条件(OUT1) :Xを2で割った余りが1 (X=1,3,5)の場合
LEDもぐら2の条件(OUT2) :Xを2で割った値をさらに2で割った余りが1(X=2,3,6,7)の場合
LEDもぐら3の条件(OUT3) :Xを4で割った値をさらに2で割った余りが1(X=4,5,6,7)の場合

※本キットではLEDが3個あるので状態の組み合わせは8通りが可能です。しかし、全てのLEDが消灯している状態(X=0)が連続するとゲームとして不自然になるので、X=0の状態は使用しないようにしています。
(少なくともどれか1匹はもぐらが出ているようにしています)

③ LEDもぐらを出したら、変数Dで指定した時間だけWAIT文で待ちます。

④ もし変数Nの値(もぐらの残り匹数)が0より大きければ、②にジャンプし、次のもぐらを出します。
もしNの値が0以下であれば、IchigoJam本体のタクトスイッチを押すまで待機し、スイッチを押したら②以下の操作を繰り返します。

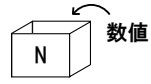
キーワードの説明

変数 D (20行)

変数とは、プログラム中で使用する数値を入れるための「箱」です。箱を区別するために、箱には「名前」をつけます。(これを「変数名」といいます)

IchigoJamのBASICでは、変数名は「A」から「Z」までのアルファベット1文字ですので、全部で26個の変数が使用できます。

コピー(代入)



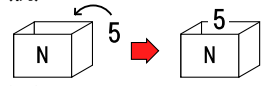
箱に式の計算結果や別の箱の中身を入れる(コピーする)ことを「代入」といいます。
(※コピー元の箱の中身は変化しません)

(参考 :BASICでは変数を使用する前に「 この名前の変数を使用します」と宣言する必要はありません)

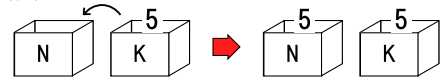
(参考2 :IchigoJamのBASICで扱える数値の範囲は、-32768から32767までの整数です)

変数に数値を代入するには、次のように書きます。

(例) N = 5

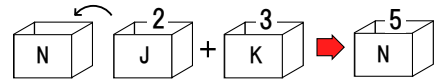


(例) N = K



足し算や引き算などの計算にも使用します。

(例) N = J + K



RND()関数 (30行)

RND()関数は0以上かっこ「()」内の数値未満のランダムな数値を返す関数です。※RND()関数のかっこ「()」内の数値は含みません。

(例) N=RND(6)+1

この例では、0から5までのランダムな数値を求め、それに1を足したものを変数Nに代入しています。

IF文と条件判断 (40行)

IF文は、条件によって処理内容を変えたいときに使います。

(例) IF BTN()=1 THEN GOTO 50

条件式　条件式が真のとき実行する文

この例では、BTN()関数が返す値が1と等しいかどうか比較し、もし真ならば(等しければ)「THEN」以下の文を実行します。(偽ならば(等しくなければ)次の行に移ります)

※「THEN」を省略して条件の後に直接実行させる文を書いてもかまいません。

条件式が偽のときに実行する文を書く必要があるときは、条件式が真のときに実行する文に続けて「ELSE」と書き、その後ろに条件式が偽のときに実行する文を書きます。

(例) IF A<B THEN M=B ELSE M=A

条件式　条件式が真のとき実行する文　条件式が偽のとき実行する文

この例ではAがBより小さいかどうか比較し、もし真ならば(A<Bならば)MにBを、偽ならば(A>=Bならば)MにAを代入します。

割り算の余り (40行)

IchigoJamのBASICでは、足し算、引き算、掛け算、割り算の四則演算に加えて、割り算の余りを求めることができます。

(例) IF (N%2)=1 THEN OUT 4,0

割り算の余りを求める演算子は「%」です。この例では、Nを2で割ったときの余りを求めて、その余りが1と等しいかどうか比較しています。(奇数か偶数かを調べています)

GOTO文(40行)

GOTO文は、指定した行番号の行にジャンプする文です。

(例) 200 GOTO 20

上の例では、プログラムの20行にジャンプします。「GOTO」に続けて、ジャンプ先の行番号を書きます。

BASICのプログラムは行番号の若い順に順次実行されますが、GOTO文はプログラムの流れを変更する文です。(プログラムの流れを変更する文には他にGOSUB文、RETURN文などがあります)

BEEP文 (50行)

BEEP文は、IchigoJam本体に接続した圧電サウンダーを鳴らす文です。

(例) BEEP 5,1

この例では、圧電サウンダーを音の高さ5、長さ1で鳴らします。BEEP文で指定する最初の数値は音の高さ(1～255、大きくなるほど音が低くなる)、2番目の数値は鳴らす長さ(1/60秒単位)です。

(例) BEEP

BEEP文の後ろの数値を省略することも可能です。(デフォルトの音の高さと長さで鳴ります)

WAIT文 (100行)

WAIT文は、プログラムの実行を指定した時間だけ一時停止させる文です。一時停止させる時間は1/60秒(約16.7ミリ秒)単位で指定します。

(例) WAIT 60

この例では、プログラムの実行を1秒(1/60秒 x 60 = 1秒)だけ一時停止させます。

一時停止させる時間として、定数の代わりに変数を含んだ式を指定することもできます。

(例) WAIT 29-M

この例のように変数を含んだ式を指定すると、待ち時間を可変にすることも可能です。

BTN()関数 (120行)

BTN()関数は、IchigoJam本体のタクトスイッチが押されていれば「1」、押されていなければ「0」を返す関数です。

(例) IF BTN()=1 THEN GOTO 50

この例ではタクトスイッチを押した場合に50行にジャンプします。

IchigoJam バージョン1.0.1では、IchigoJam本体のタクトスイッチの状態以外にキーボードの状態も取り込めます。状態を調べたいキーは、BTN()関数のかっこの中に指定します。

BTN(SPACE):スペースキーの状態
BTN(UP):上矢印キーの状態
BTN(DOWN):下矢印キーの状態
BTN(LEFT):左矢印キーの状態
BTN(RIGHT):右矢印キーの状態

かっこの中身を省略した場合はIchigoJam本体のタクトスイッチの状態を取り込みます。(バージョン0.9.7と同じ動作をします)

4 もぐらたたきゲームの原型を作ろう

前の章でLEDもぐらをランダムに出現させましたので、次に出現したLEDもぐらをブレッドボード上のタクトスイッチでやっつけられるようにし、もぐらたたきゲームの原型を作りましょう。

プログラムリスト

```
10 CLS:PRINT "ﾓｸﾞﾗ ﾀﾀｷ ｹﾞｰﾑ"
20 OUT 0:D=60:N=25
30 X=RND(7)+1
40 IF (X%2)=0 OUT 1,0:GOTO 60
50 BEEP 10,5:OUT 1,1:N=N-1
60 IF ((X/2)%2)=0 OUT 2,0:GOTO 80
70 BEEP 12,5:OUT 2,1:N=N-1
80 IF ((X/4)%2)=0 OUT 3,0:GOTO 100
90 BEEP 15,5:OUT 3,1:N=N-1
100 CLT
110 IF (X%2)=0 GOTO 140
120 IF IN(1) GOTO 140
130 OUT 1,0:X=X&6:BEEP 10,20
140 IF ((X/2)%2)=0 GOTO 170
150 IF IN(2) GOTO 170
160 OUT 2,0:X=X&5:BEEP 12,20
170 IF ((X/4)%2)=0 GOTO 200
180 IF IN(4) GOTO 200
190 OUT 3,0:X=X&3:BEEP 15,20
200 IF TICK()<D GOTO 110
210 OUT 0
220 WAIT D:IF N>=0 GOTO 30
230 BEEP 30,30
240 IF BTN()=0 GOTO 240
250 GOTO 20
```

キーワード

CLS文　　TICK()関数とCLT文

プログラムの説明

(1) 遊び方

プログラムをRUNすると、前の章と同じようにランダムにLEDもぐらが出現します。

出現したLEDもぐらをやっつけるには、出現したLEDもぐらのところのタクトスイッチを押します。LEDもぐらをやっつけると、「ピー」と音がしてLEDが消灯します。

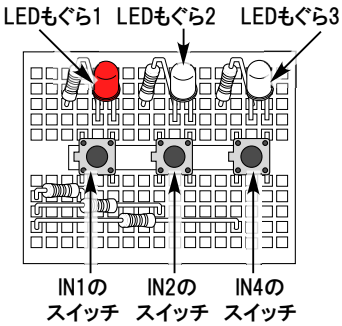
出現するもぐらの総数は25匹です。(プログラム20行の変数Nの値を変更するともぐらの総数を変更できます)

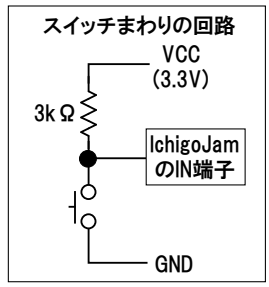
ゲーム終了後、IchigoJam本体のタクトスイッチを押すと再び新しくゲームをはじめることができます。

> 本キットのプログラムではスイッチを押しっぱなしにした場合の対策はしていません。(プログラムサイズの制約があるため)
>
> スイッチを3個とも押しっぱなしにしておくと出現したもぐらを全部やっつけることができますが、そのような行為はしないようにしましょう。

(2) プログラムの動作について

ランダムにLEDもぐらを出現させる機能については前の章の説明を参照してください。ここではスイッチを押して出現したLEDもぐらをやっつける処理について説明します。





本キットのスイッチまわりの回路は右上の図のようになっています。スイッチを押していないときはIchigoJamのIN端子は3kΩの抵抗を介して3.3Vの電源(VCC)につながっています。(IN()関数で状態を読み取ると「1」が返ります)

スイッチを押すとIchigoJamのIN端子はグラウンド(0V)につながり、IN()関数で状態を読み取ると「0」が返ります。

LEDもぐらを出現させたあと、スイッチ読み取りループに入ります。スイッチ読み取りループの中では制限時間の間繰り返してスイッチの状態をチェックし、LEDもぐら1が出現しているときはIN1のスイッチ、LEDもぐら2が出現しているときはIN2のスイッチ、LEDもぐら3が出現しているときはIN4のスイッチが押されていたならばもぐらをやっつけたとしてブザーを鳴らします。

単純にWAIT文で時間待ちしたあとスイッチの状態をチェックする方法をとっていないのは、LEDもぐらが複数匹出現したときスイッチをどの順番で押してももぐらをやっつけることができるようにするためです。

スイッチ読み取りループを1回回るごとにTICK()関数で経過時間を調べ、あらかじめ設定してある制限時間を超えたら時間切れとして次のもぐらを出す処理を行います。

キーワードの説明

CLS文 (10行)

CLS文は、画面の表示をクリアする(何も表示されていない状態にする)ためのコマンドです。

(例)CLS

CLS文がクリアするのは画面の表示だけです。プログラムや変数の値などはクリアされません。

CLT文 (100行)とTICK()関数 (200行)

IchigoJam内部には、プログラムの実行とは関係なく約1/60秒(16.67ミリ秒)ごとにカウントアップするTICKカウンタがあります。

このTICKカウンタはTICK()関数でいつでも読み取ることができます。

(例)N=TICK()

上の例はIchigoJam内部のTICKカウンタを読み取り、その読み取り値を変数Nに代入しています。

(例) CLT

TICKカウンタの値はCLT文でいつでも0に戻すことができます。

CLT文でTICKカウンタを0に戻し、次にTICK()関数で内部カウンタの値を読み取ることで、CLT文を実行してからTICK()関数を呼び出すまでの間の経過時間をカウント数として知ることができます。

(参考) プログラムRUN時の内部カウンタの値

IchigoJam バージョン0.9.7では、プログラムをRUNしたときに自動的に内部カウンタの値を0に戻すようになっています。(CLT文でも戻せます)

IchigoJam バージョン1.0.1では、内部カウンタの値はプログラムをRUNしても0に戻りません。(戻すにはCLT文を実行する必要があります。

(参考) TICK()関数の最大値

IchigoJam バージョン1.0.1では、TICK()関数で得られるカウンタの最大値は32767です。(バージョン0.9.7では16383)

(どちらのバージョンでも、最大値までカウントアップしたあと、一巡して値が0になります)

本キットのプログラムでは、最大値までカウントアップすることがないので、どちらのバージョンでも問題なく使用できます。

---

5 もっとゲームらしくしよう

上ページで作成したもぐらたたきゲームの原型をもっとゲームらしく改良して、楽しく遊べるようにしましょう。

出現したLEDもぐらをやっつけると点数(SCORE)が加算されるようにします。また、もぐらをやっつけると次第にゲームのスピードが速くなり、難易度がアップする仕掛けにします。

1ゲームが終了すると、獲得した点数に応じてあなたの腕前がモニタ画面に表示されます。

プログラムリスト

```
10 CLS:PRINT "ﾓｸﾞﾗ ﾀﾀｷ"
15 PLAY "DBGDBGDBDG":WAIT 360
20 OUT 0:D=60:N=25:S=0
30 X=RND(7)+1:LC 0,2:PRINT "SCORE : ";S
35 IF N<=0 THEN GOTO 220
40 IF (X%2)=0 OUT 1,0:GOTO 60
50 BEEP 10,5:OUT 1,1:N=N-1
60 IF ((X/2)%2)=0 OUT 2,0:GOTO 80
70 BEEP 12,5:OUT 2,1:N=N-1
80 IF ((X/4)%2)=0 OUT 3,0:GOTO 100
90 BEEP 15,5:OUT 3,1:N=N-1
100 CLT
110 IF (X%2)=0 GOTO 140
120 IF IN(1) GOTO 140
130 OUT 1,0:X=X&6:BEEP 10,20
135 S=S+1:D=D-2
140 IF ((X/2)%2)=0 GOTO 170
150 IF IN(2) GOTO 170
160 OUT 2,0:X=X&5:BEEP 12,20
165 S=S+1:D=D-2
```

ページ右上に続きます

プログラムの続きです

```
170 IF ((X/4)%2)=0 GOTO 200
180 IF IN(4) GOTO 200
190 OUT 3,0:X=X&3:BEEP 15,20
195 S=S+1:D=D-2
200 IF TICK()<D GOTO 110
210 OUT 0:WAIT D:GOTO 30
220 LC 0,4:PRINT "GAME OVER"
230 LC 0,6
240 PLAY "O4T180"
250 IF S<25 GOTO 290
260 PRINT "ｵﾐｺﾞﾄ!!"
270 PLAY ">C<BAG>C<BAG>CED"
280 GOTO 390
290 IF S<20 GOTO 330
300 PRINT "ﾅｶﾅｶ ﾔﾙﾅ!"
310 PLAY "CEG>C"
320 GOTO 390
330 IF S<10 GOTO 370
340 PRINT "ﾏｱﾏｱ ｶﾅ?"
350 PLAY "AF"
360 GOTO 390
370 PRINT "ｳｰﾑ ｻﾞﾝﾈﾝ"
380 PLAY "A"
390 IF BTN()=0 GOTO 390
400 GOTO 10
```

キーワード

LOCATE(LC)文　　PLAY文

プログラムの説明

遊び方については前の章のプログラムと同じです。

前の章のプログラムに次の機能を追加しました。

① 点数(もぐらをやっつけた数)をカウントするための変数(S)を追加しました。

② もぐらを1匹やっつける度に変数Dの値を1ずつ減らし、もぐらを出したあとスイッチの状態を判別するまでの待ち時間がすこしずつ短くなるようにしました。

② ゲーム終了時に点数(S)の値に応じてPLAY文で簡単なメロディーを鳴らし、モニタ画面に腕前を表示するようにしました。

キーワードの説明

LOCATE文 (30行) ※省略形 LC

LOCATE文は、次に実行するPRINT文の画面上の表示位置を指定するのに使います。

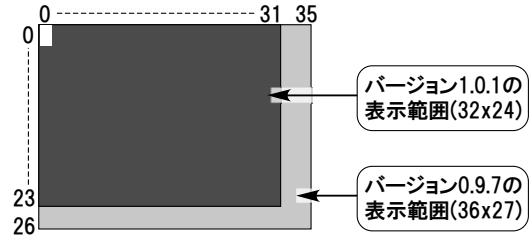
表示する位置は、LOCATEの後ろに「横の位置」「縦の位置」の順で指定します。表示位置は横の位置は画面左端、縦の位置は画面上端を「0」として数えます。
(※「1」からではありません。間違えやすいので注意してください)

(例) LOCATE 0,1

この例は、次に実行するPRINT文の表示位置を、画面の2行目の一番左端(1文字目)にします。

LOCATEの代わりにLCという省略形も使用できます。

(例) LC 0,1



LOCATE文で指定できる「横の位置」「縦の位置」の最大値(画面サイズ)は、IchigoJamのバージョンにより異なりますのでプログラムを書く際注意が必要です。バージョンにより正常に表示されないことがあります。(エラーにはなりません)

> バージョン0.9.7の画面サイズは36文字x27行、バージョン1.0.1の画面サイズは32文字x24行になっています。

PLAY文 (15行)

PLAY文は、IchigoJamのサウンド出力機能を利用して簡単なメロディを演奏するための文です。演奏中もプログラムの実行は停止しません。

> ※ 和音は演奏できません。また、半音(ピアノの黒い鍵盤の音)は出ません。

(例) PLAY "CDE"

上の例は、IchigoJamに「ドレミ」を出力させる例です。上の例のように、演奏させるメロディをアルファベットで表します。

音の長さは特に指定しない限り4分音符(1分あたり120)になります。

アルファベットと音の高さは次のように対応します。

ド : C　レ : D　ミ : E　ファ : F
ソ : G　ラ : A　シ : B　休符 : R

音の長さは音の高さを表すアルファベットの後ろに1/2/4/8/16/32の数字で指定します。長さを指定しないときは4分音符になります。後ろにピリオド(.)を書くことで音の長さを半分だけ伸ばすことも可能です。

(例) PLAY "C4D8E8."

上の例では、ドの4分音符、レの8分音符、ミの付点8分音符を演奏します。

オクターブ上げる/下げるの指定も可能です。「>」の後はオクターブ上がり、「<」の後はオクターブ下がります。

(例) PLAY "CDEFGAB>C"

上の例は「ドレミファソラシド」になります。

> PLAY文で指定できる内容一覧はIchigoJamに一緒に入っているBASIC命令一覧表の裏に載っていますので、参照してください。