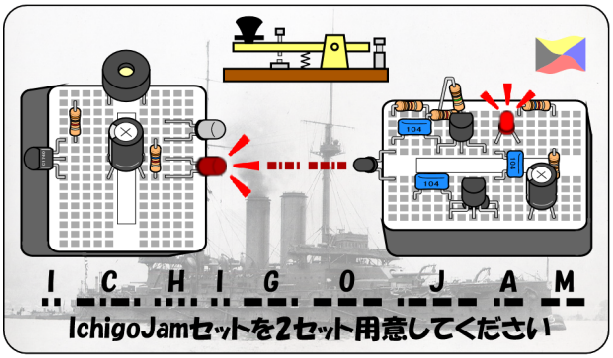
IchigoJam 電子工作パーツセット
赤外線モールス通信機
MORSE-jamP　送信機 & 受信機セット

Infrared Morse TX & RX

IchigoJam 0.9.7 / 1.0.1対応

0 概要



電気通信の歴史上重要なものに、モールス通信があります。
本キットでモールス通信を楽しみながらモールス符号を覚えましょう。
IchigoJamを2セット使用することで赤外線によるモールス通信の実験ができます。

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。
和文モールスには対応していません。
手打ちのモールス符号は正常に解読できないことがあります。(速度が一定のため)

送信機部品表

	品名/型番/値	数量	備考
1	赤外線LED	1	
2	LED 白	1	
3	トランジスタ 2SC1740	1	
4	1/4W抵抗 47Ω	1	黄紫黒金の色帯
5	1/4W抵抗 1.6kΩ	1	茶青赤金の色帯
6	1/4W抵抗 3kΩ	1	橙黒赤金の色帯
7	電解コンデンサ 220μF	1	
8	ジャンプワイヤ 10cm	5	
9	ブレッドボード	1	

※IchigoJamに付属の圧電サウンダーが別途必要です。

受信機部品表

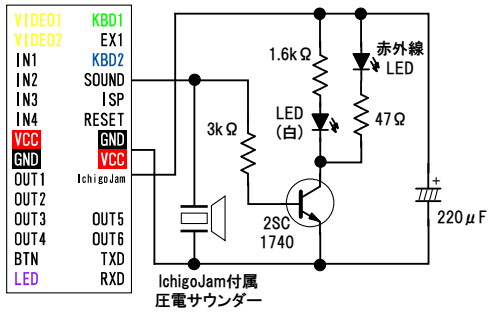
	品名/型番/値	数量	備考
1	フォトトランジスタ	1	
2	トランジスタ 2SC1740	2	
3	LED 赤	1	
4	ダイオード	1	
5	1/4W抵抗 180Ω	1	茶灰茶金の色帯
6	1/4W抵抗 1.6kΩ	1	茶青赤金の色帯
7	1/4W抵抗 3kΩ	2	橙黒赤金の色帯
8	1/4W抵抗 1MΩ	1	茶黒緑金の色帯
9	積層セラミックコンデンサ 0.1μF	3	104の表記
10	電解コンデンサ 220μF	1	
11	ジャンプワイヤ 10cm	9	
12	ブレッドボード	1	

※部品表は予告なく変更することがあります。

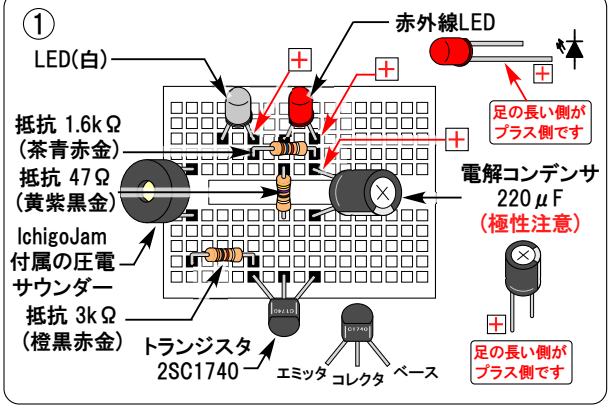
※ブレッドボードの色は特に決まっていません。

1 送信機の組み立てかた

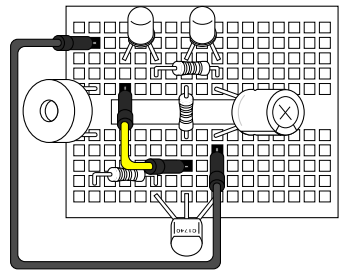
回路図 ※予告なく変更することがあります



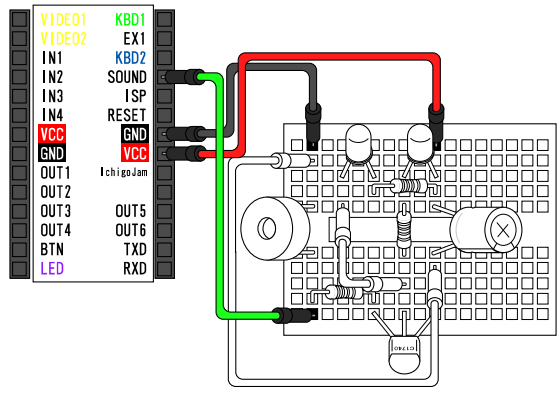
①



②

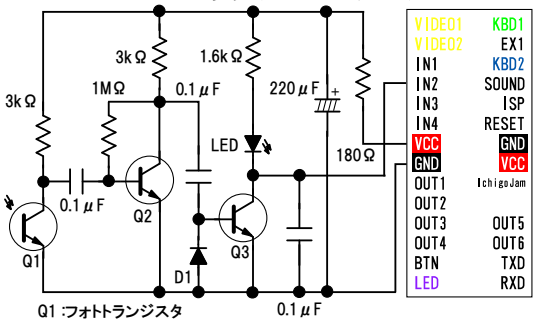


③ IchigoJamとの接続



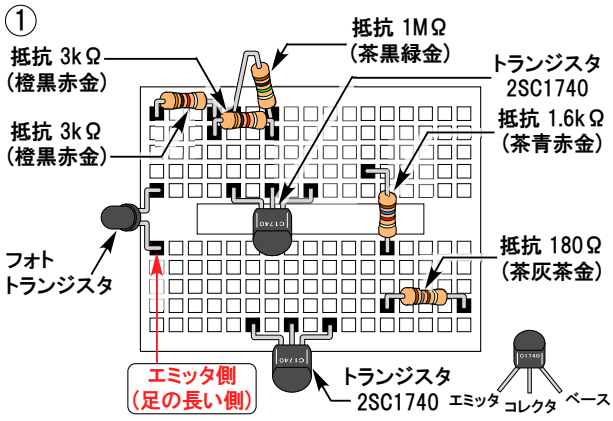
受信機の組み立てかた

回路図 ※予告なく変更することがあります



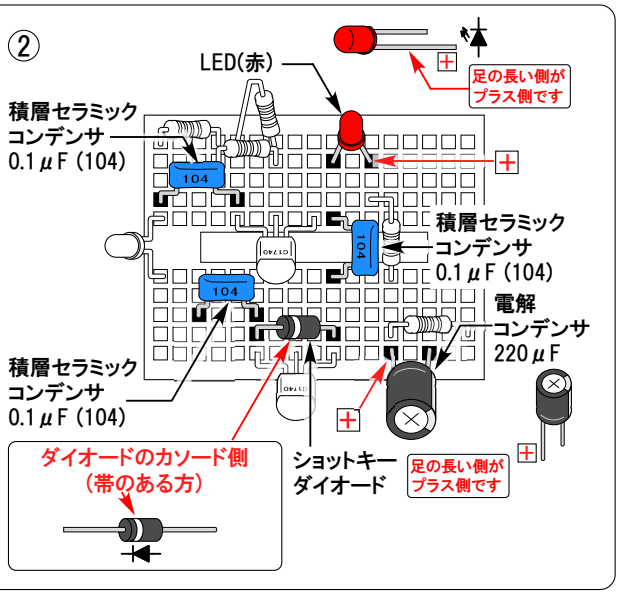
Q1 :フォトトランジスタ
Q2, Q3 :トランジスタ 2SC1740
D1 :ショットキーダイオード

①



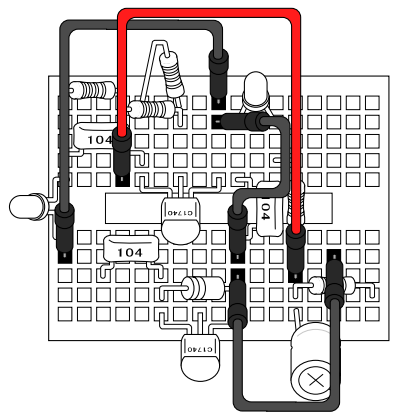
フォトトランジスタは足の長いほうがエミッタ側(回路のマイナス側)です。

②

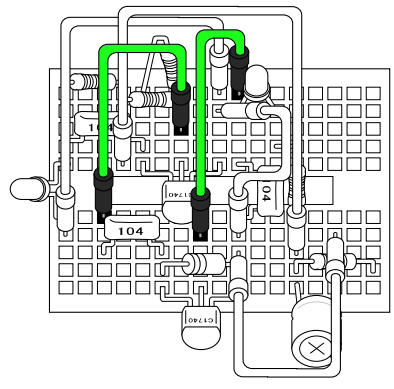


ページ右上③に続きます

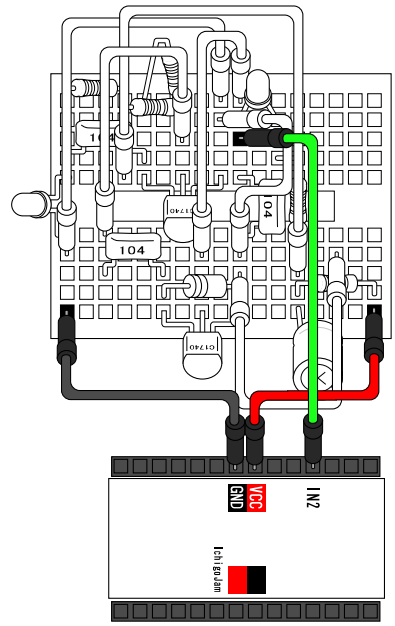
③



④



⑤ IchigoJamとの接続



プログラムのセーブとロードのしかた

プログラムを作成したら、SAVEコマンドでプログラムを保存します。

```
    ⋮
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。

はじめのプログラム

組み立てた回路の動作チェックを兼ねて、簡単なモールス送受信のプログラムを作成します。

送信側は試験符号(VVVの連続)をIchigoJamのSOUND端子に出力します。受信側はフォトトランジスタで受信した信号を画面に表示します。

送信側プログラムリスト

```
10 PRINT "ﾃｽﾄ ｿｳｼﾝ ﾌﾟﾛｸﾞﾗﾑ"
20 X=23
30 IF X%2=1 THEN GOTO 70
40 PRINT "-";:BEEP 10,15:WAIT 15
50 X=(X-2)/2
60 GOTO 90
70 PRINT ".";:BEEP 10,5:WAIT 5
80 X=(X-1)/2
90 WAIT 5:IF X>0 THEN GOTO 30
100 PRINT:WAIT 15:GOTO 20
```

受信側プログラムリスト

```
10 PRINT "ﾓｰﾙｽ ｼﾞｭｼﾝ ﾌﾟﾛｸﾞﾗﾑ"
20 L=0:H=0:J=1:K=1
30 WAIT 1
40 K=J:J=IN(2)
50 IF J<>K THEN GOTO 90
60 IF J=0 THEN L=L+1 ELSE H=H+1
70 GOTO 30
80 REM ﾌｺﾞｳ ｶｲｾｷ
90 IF H>0 THEN GOTO 140
100 IF L>20 THEN PRINT "-";:GOTO 120
110 IF L>5 THEN PRINT "."
```

ページ右上に続きます

プログラムの続きです

```
120 L=0:H=0
130 GOTO 30
140 IF H>80 THEN PRINT " ";:GOTO 160
150 IF H>20 THEN PRINT "/";
160 L=0:H=0
170 GOTO 20
```

```
    ⋮
80 GOTO 30

SAVE0↵
```

プログラムを書いたあと「SAVE0↵」でプログラムを保存してください。
(↵は[ENTER]キーです)
※0番に大事なプログラムが入っている場合は、他の番号を指定して保存してください。

キーワード

- PRINT文の使い方
- IF文と条件判断
- BEEP文
- IN()関数
- 変数 X
- 割り算の余り
- WAIT文
- REM文(コメント)

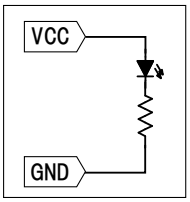
部品と回路の説明

(1) 赤外線LEDについて

赤外線LEDは普通の赤や緑などのLEDと同じ、電流を流すと光るLED(発光ダイオード)の仲間です。普通のLEDは赤や緑などの目に見える波長の光を出しますが、赤外線LEDは目に見えない赤外線を出します。

肉眼で見る限り、赤外線LEDは電流を流していても光っているように見えませんが、デジカメなどで写すと左のように光っているのがわかります。(最近のデジカメでは写らないものもあるようです)

赤外線LEDはテレビやエアコンなどのリモコンなど情報を送ったり、パソコンのプリンタの紙検出など物体のあるなしを検出したりするために幅広く使用されています。

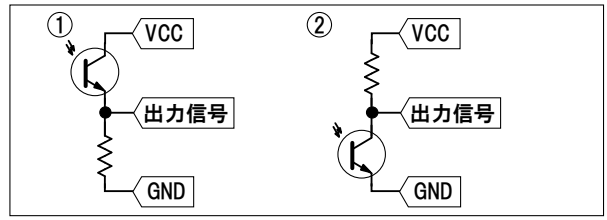


赤外線LEDの使用方法は普通のLEDと同じです。LEDと直列に抵抗を入れて流れる電流を適宜制限します。トランジスタで電流をON/OFFすると赤外線のON/OFFができます。

(2) フォトトランジスタについて

フォトトランジスタはトランジスタの仲間です。光が当たるとコレクタ電流が流れ、光の強弱を電気信号として取り出すことができます。感度が高く使いやすいので、センサとして物体のあるなしを検出したりするのに広く使用されています。

フォトトランジスタには無色透明のパッケージに入ったものと、黒い樹脂パッケージに入ったものがあります。黒い樹脂パッケージのものは室内の明かりなど周囲の光に邪魔されずに赤外線のみを受けるのに使用されます。



フォトトランジスタを使用するときは上図のように接続します。フォトトランジスタのデータシートには左側①の回路で載っていることが多いのですが、本キットではわかりやすくするため右側②の回路を使用しています。

フォトトランジスタに光が当たるとコレクタ電流が流れ、負荷の抵抗の働きで光の変化を電圧の変化として取り出すことができます。

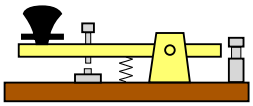
(3) 本キットの回路について

本キットの回路では、IchigoJamのSOUND出力をモールス符号に合わせてON/OFFさせ、赤外線で送信しています。これは周囲の光に邪魔されにくいようにするためと、受信側の処理を楽にするためです。(トーンの周波数は異なりますが、皆さんの周囲にもあるテレビなどのリモコンも同じ方式です)

送信部からの光信号を受信すると、フォトトランジスタから電気信号が出力されます。この信号を増幅したあと検波回路に通してモールス符号に合わせた電気信号を得ています。

プログラムの説明

(1) モールス通信について

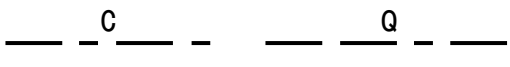


モールス通信は電流や光、音などを断続して文字を送る通信方式で、アメリカの発明家サミュエル・モールスにより1844年に発明されました。

上の絵はモールス通信に使用されている電鍵(キー)です。機能的にはただのスイッチです。手でつまみを操作して電流をON/OFFすることで信号を送ります。

モールス通信は電流に限らず光や音など、ON/OFFさえできればどんなものでも通信手段として使えること、送信と受信の機械が単純なものですむことから、業務用や非常用の通信手段として幅広く使用されました。(最近では業務用にはあまり使用されていません)

モールス通信に使用する符号をモールス符号といいます。モールス符号は短点と長点の組み合わせでできていて、次のような約束事があります。



(1) 長点の長さは短点3個分
(2) 短点、長点の間のスペースは短点1個分あける
(3) 単語中文字間のスペース(上の例ではCとQの間)は短点3個分あける
(4) 単語と単語の間のスペースは短点7個分あける

モールス符号の特色は符号の長さが文字によって異なることです。(このようなタイプの符号を可変長符号といいます)

この説明書の最後に欧文のモールス符号表を載せていますので、参考にしてください。

欧文のモールス符号の場合、「E」や「T」、「A」などのよく出現する文字には短い符号を、あまり出現しない文字には長い符号を割り当てることで効率よく通信できるよう工夫されています。

参考 (和文のモールス符号について)

本キットでは扱いませんが、和文(日本語のカタカナ)のモールス符号も存在します。

通信スピードは特に決められていません。電波状態や通信する人の技量などによって適宜加減していました。(実際にはちょっと速い程度がリズミカルで聞き取りやすいようです)

(2) 送信側のプログラムについて

送信側のプログラムは、送信機試験用の符号(VVV)を繰り返し赤外線で送信するプログラムです。

送信側のプログラムをRUNすると、送信側ブレッドボードから赤外線で「・・・ー・・・ー・・・ー」　という信号(※)が送信されます。もし手元にデジカメなどがありましたら、赤外線LEDの部分をデジカメなどのモニター画面で観察してみてください。赤外線LEDが符号通り点滅しているのが見えると思います。

「VVV」符号について

「・・・ー・・・ー・・・ー」　という信号はモールス符号の「VVV」で、送信の試験を行うときに打つ符号として国際的に決まっています。ほかにもモールス通信にはいろいろな略号があります。巻末の参考資料を見てください。

モールス符号を発生させる方法として、本キットのプログラムでは次の方法を利用しています。

(1) あらかじめ箱(変数)のなかにある数値が入っているとします。

(2a) 箱の数値が奇数の場合、短点を出力し、数値から1を引いて2で割ります。

(2b) 箱の数値が偶数の場合、長点を出力し、数値から2を引いて2で割ります。

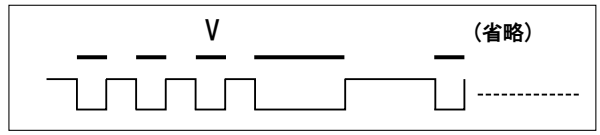
(3) 箱の数値が0になるまで(2a)/(2b)を繰り返します.

(4) 箱の数値が0であれば1文字送信終わりなので、短点3個分のスペースをあけます。

(3) 受信側のプログラムについて

送信側のプログラムをRUNした状態で、受信側のプログラムをRUNすると、受信した符号に合わせてブレッドボード上の赤LEDが点滅し、画面に短点「・」と長点「ー」が表示されます。

ブレッドボード上の赤LEDがうまく点滅するように、送信側ブレッドボードと受信側ブレッドボードの向きを調節してください。



本キットの回路では、光信号を受信するとIchigoJamのIN2端子が「L」になります。プログラムでは、符号の短点のスピードより十分短い周期でIN2端子の状態を監視し、「L」の時間の長さと「H」の時間の長さを測ります。

「L」の時間の長さが長点の判別基準の長さ以上であれば長点と認識し、画面に「ー」を表示します。短点の判別基準の長さ以上であれば短点と認識し、画面に「・」を表示します。

「H」の時間の長さが単語の判別基準の長さ以上であれば単語の切れ目と認識し、画面に空白文字を表示します。長点の判別基準の長さ以上であれば文字の間と認識し、画面に「/」を表示します。

キーワードの説明

PRINT文の使い方 (送信10行/受信10行)

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。

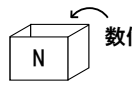
(例) PRINT W/10;″.″;W%10;″V″
この例ではまず変数Wの値を10で割った値(小数点以下は切捨てとなります)を表示し、そのあとに小数点記号(ピリオド「.」記号)を表示し、そのあとに変数Wの値を10で割った余りを表示、最後に文字「V」を表示します。

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。

変数 X (送信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)



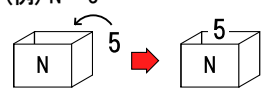
箱に式の計算結果や別の箱の中身を入れる(コピーする)ことを「代入」といいます。
(※コピー元の箱の中身は変化しません)

(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)

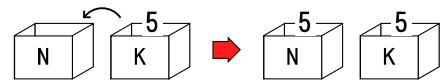
(参考2 :IchigoJamのBASICで扱える数値の範囲は、-32768から32767までの整数です)

変数に数値を代入するには、次のように書きます。

(例) N = 5



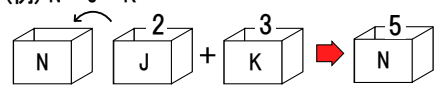
(例) N = K



※コピー元の変数(K)の値は変化しません

足し算や引き算などの計算にも使います。

(例) N = J + K



IF文と条件判断（送信30行/受信50行）

IF文は、条件によって処理内容を変えたいときに使用します。

（例）IF V>=186 THEN OUT 1,1

条件式　条件式が真のとき実行する文

この例では、Vの値が186以上かどうか比較し、もし真ならば(以上ならば)「THEN」以下の文を実行します。(偽ならば(未満ならば)次の行に移ります)

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 V>=186 THEN OUT 1,1 ELSE OUT 1,0

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Vの値が186以上かどうか比較し、もし真ならば(V>=186ならば)OUT1端子に「1」を、偽ならば(V<186ならば)OUT1端子に「0」を出力します。

割り算の余り（送信30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)

BEEP文（送信40行）

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

（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（送信40行/受信30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。

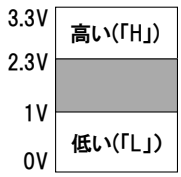
一時停止させる時間として、定数の代わりに変数を含んだ式を指定することもできます。

（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。

IN()関数（受信40行）

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。

REM文(コメント)（受信80行）

REM文は、プログラムにコメント(メモ)を入れるために使います。

（例）REM ｽｲｯﾁ ｦ ｵｽﾏﾃﾞ ﾏﾂ

「REM」の後ろはプログラムの実行時は無視されます。

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。

入門プログラム

キーボードから入力した文字をモールス符号で送信します。受信側ではモールス符号を画面に表示するとともに、あとの「上級プログラム」での解読に利用する解読用コードを表示します。

送信側プログラムリスト

```
10 PRINT "ﾓｰﾙｽ ｿｳｼﾝ ﾌﾟﾛｸﾞﾗﾑ"
20 LET [30],0,0,0,0,81
30 LET [35],0,0,0,93,0
40 LET [40],44,108,0,41,114
50 LET [45],96,105,40,62,61
60 LET [50],59,55,47,31,32
70 LET [55],34,38,46,70,0
80 LET [60],0,48,0,75,85
90 LET [65],5,16,20,8,1
100 LET [70],19,10,15,3,29
110 LET [75],12,17,6,4,14
120 LET [80],21,26,9,7,2
130 LET [85],11,23,13,24,28,18
140 REM ﾓｼﾞ ヲ ｺｰﾄﾞ ﾆ ﾅｵｽ
150 A=INKEY()
155 IF A=0 THEN GOTO 150
160 IF A<34 THEN GOTO 150
165 PRINT CHR$(A);" ";
170 X=[A]
180 IF X=0 THEN GOTO 150
190 REM ﾌｺﾞｳ ヲ ｿｳｼﾝ
200 IF (X%2)=1 THEN GOTO 250
210 PRINT "-";
220 BEEP 10,15:WAIT 15
230 X=(X-2)/2
240 GOTO 280
```

ページ右上に続きます

プログラムの続きです

```
250 PRINT ".";
260 BEEP 10,5:WAIT 5
270 X=(X-1)/2
280 WAIT 5
290 IF X>0 THEN GOTO 200
300 WAIT 15
310 PRINT:GOTO 150
```

受信側プログラムリスト

```
10 PRINT "ﾓｰﾙｽ ｼﾞｭｼﾝ ﾌﾟﾛｸﾞﾗﾑ"
20 L=0:H=0:J=1:K=1:X=0
30 WAIT 1
40 K=J:J=IN(2)
50 IF J<>K THEN GOTO 90
60 IF J=0 THEN L=L+1 ELSE H=H+1
70 GOTO 30
80 REM ﾌｺﾞｳ ｶｲﾄﾞｸ
90 IF H>0 THEN GOTO 170
100 IF L<=20 THEN GOTO 130
110 PRINT "-";:X=X*2+2
120 GOTO 150
130 IF L<=5 THEN GOTO 150
140 PRINT ".";:X=X*2+1
150 L=0:H=0
160 GOTO 30
170 IF H<=80 THEN GOTO 200
180 PRINT " ";X
190 GOTO 220
200 IF H<=20 THEN GOTO 220
210 PRINT " ";X
220 L=0:H=0
230 GOTO 30
```

キーワード

- 配列とLET文
- INKEY()関数
- CHR$()関数

プログラムの説明

(1) 送信側プログラムの処理

「はじめのプログラム」で実験したモールス符号送信アルゴリズムを応用して入力した文字をモールス符号に変換、送信します。

プログラムをRUNするとキー入力待ちになりますので、キーボードから何か文字を入力してください。画面にモールス符号が表示され、ブレッドボード上の赤外線LEDからモールス符号が送信されます。

プログラムの冒頭でモールス符号発生用コードを配列に格納します。
メインループではキーボードからの文字入力を読み取り、文字コードに対応した符号発生用コードを配列から取得します。
配列から取得した符号発生用コードを利用して、モールス符号の生成を行います。

(2) 受信側プログラムの処理

「はじめのプログラム」ではモールス符号の短点と長点、文字の切れ目と単語の切れ目を認識しました。

認識した短点と長点、文字の切れ目を利用してモールス符号を解読用コードに変換します。

モールス符号を解読用コードに変換する手続きは次のようになっています。

(1) 解読用コード用の変数(X)を0にクリアします。(20行)

(2a) 長点を受信したら変数Xの値を2倍して2を足します。(110行)

(2b) 短点を受信したら変数Xの値を2倍して1を足します。(140行)

(3) 短点3個分以上の空白が来たら符号の終わりとみなして変数Xの値を表示します。(180行/210行)

(4) (1)に戻り新しい符号を受信します。

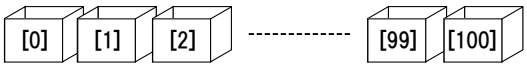
キーワードの説明

配列とLET文 (送信20行)

プログラム中で同じ性質を持った沢山のデータを扱いたいことがよくあります。代表的なものとしては、次のようなものが考えられます。

(1) クラス全員の試験の成績
(2) 文字コードの変換表

「配列」とはこうしたデータを扱うとき便利なように変数を沢山並べたものです。



何番目の箱を使用するかはかっこ([ ])の中の数字(添え字といいます)で指定します。IchigoJamの場合、0番目から100番目まで全部で101個の箱が指定できます(※)。

(※ 添え字は0から100までの範囲です。この範囲を超えるとエラーメッセージを出して停止します。)

配列を使用するとき、0番目の箱から順番に使用する必要はありません。途中から使用してもかまいません。

(例) X=[5]

上の例は配列の5番目の箱の中身を変数Xにコピーします。

(例) X=[K]

この例のように、添え字として変数名を使用すると、変数の値で使用する箱を選ぶことができます。

LET文は変数や配列に値を入れるための代入文です。

(例) LET A,1

上の例は変数Aに値1をコピーします。(「A=1」と書くのと同じ操作になります)

変数に値を入れるときはLET文を使用するよりも普通に式で書いたほうがわかりやすいため、LET文はあまり使用されません。

しかし、配列の連続する複数個の箱に値をまとめて入れたい場合は、次のようにLET文で値を入れます。

(例) LET [30],0,0,0,0,81

上の例は配列の30番目の箱から順番に、「0、0、0、0、81」という値を連続して入れます。

INKEY()関数 (送信150行)

INKEY()関数は、IchigoJamのキーボードからの入力文字の文字コードを返す関数です。キーボードのキーが何も押されていないときは「0」を返します。

(例) A=INKEY()

上の例では変数Aに入力文字の文字コードを代入します。キーが何も押されていないときはAには「0」が入ります。

※ 実際にはINKEY()関数はキーボードのキー入力を直接チェックしているのではなく、IchigoJam内部にいったん蓄えられた入力文字のコードを返しています。
このため、プログラムの作り方によってはキー入力とINKEY()関数が返す文字コードが一致しないことがあります。ゲームなどに応用するときは注意してください。

CHR$()関数 (送信165行)

CHR$()関数は、指定した値を文字コードとみなし、指定した値に対応する文字を返す関数です。

(例) PRINT CHR$(A)

上の例は変数Aの値を文字コードとみなし、変数Aの値に対応する文字を表示します。

参考 (IchigoJamでの文字の扱い)

IchigoJamのBASICには文字を扱うために次のような関数が用意されています。

◎ INKEY()関数 :キーボードから入力された文字の文字コードを返します。

◎ ASC()関数 :指定した文字に対応する文字コードを返します。

◎ CHR$()関数 :指定した文字コードに対応する文字を返します。

※ 文字コード :コンピュータ上で使用する文字のそれぞれを数字で表したものです。(コンピュータ上では文字も文字コードという数字で扱われます)
たとえば、文字「A」の文字コードは65番です。

4 上級プログラム

入門プログラムで作成した受信符号解析ルーチンを利用して、受信したモールス符号を解読して文字としてモニタ画面に表示します。

送信側のプログラムは入門プログラムと同じものを使用します。

受信側プログラムリスト

```
10 CLS:PRINT "ﾓｰﾙｽ ｼﾞｭｼﾝ"
20 LET [0],69,84,73,65,78
30 LET [5],77,83,85,82,87
40 LET [10],68,75,71,79,72
50 LET [15],86,70,0,76,0
60 LET [20],80,74,66,88,67
70 LET [25],89,90,81,0,0
80 LET [30],53,52,0,51,0
90 LET [35],0,0,50,0,0
100 LET [40],43,0,0,0,0
110 LET [45],49,54,61,47,0
120 LET [50],0,0,40,0,55
130 LET [55],0,0,0,56,0
140 LET [60],57,48,0,0,0
150 LET [65],0,0,0,0,0
160 LET [70],0,0,0,0,63
170 K=1
180 LC 0,23:X=0
190 K=J:J=IN(2)
200 IF K=J GOTO 190
210 IF J=1 GOTO 280
220 WAIT 7
```

ページ右上に続きます

プログラムの続きです

```
230 IF IN(2)=0 GOTO 260
240 X=X*2+1:PRINT ".";
250 GOTO 190
260 X=X*2+2:PRINT "-";
270 GOTO 190
280 CLT
290 K=J:J=IN(2)
300 IF TICK()>6 GOTO 325
310 IF K=J GOTO 290
320 GOTO 220
325 LC 8,23
330 IF X>80 GOTO 370
340 IF X=0 THEN PRINT " ERR":GOTO 450
350 A=[X-1]
360 PRINT " ";CHR$(A):GOTO 450
370 IF X=81 PRINT " ";CHR$(34)
380 IF X=84 PRINT " ."
390 IF X=89 PRINT " @"
400 IF X=93 PRINT " '"
410 IF X=96 PRINT " -"
420 IF X=108 PRINT " )"
430 IF X=114 PRINT " ,"
440 IF X=119 PRINT " ";CHR$(58)
450 SCROLL 0:GOTO 180
```

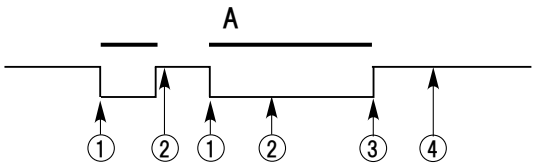
キーワード

- CLS文
- LOCATE文
- CLT文とTICK()関数
- SCROLL文

プログラムの説明

送信側からのモールス符号を受信すると、画面にモールス符号と解読された文字を表示します。

入門プログラムの符号解析ルーチンでは次の文字の符号が来ないと前の文字の受信完了がわからず、解読が間に合わないため、短点と長点の取り込みを次のように工夫しています。



① IN(2)入力が「1」から「0」に変化したら、短点の時間+α だけ待ちます

② IN(2)入力が「1」ならば短点、「0」ならば長点とみなします

③ IN(2)入力が「0」から「1」に変化したら、IchigoJam内部のTICKタイマを0にクリアします。

④ IN(2)入力が「1」から「0」に変化するのを待ちますが、このときTICKタイマの値が短点の時間+α以上になれば、その時点で符号の取り込みを打ち切り、1文字の受信が完了したとします。

1文字分の符号の取り込みが完了したら、配列を検索して、解読用コードに対応した文字を画面に表示します。(配列にはプログラムの最初で解読用コードに対応した文字コードを入れておきます)

キーワードの説明

CLS文 (10行)

CLS文は、モニタの画面表示をクリアする文です。

(例) CLS

CLS文がクリアするのはモニタの画面表示のみです。プログラムや変数の内容はクリアされません。

LOCATE文 (180行)

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

参考 (IchigoJamの画面の大きさ)

IchigoJamバージョン1.0.1の画面の大きさは32文字×24行です。バージョン0.9.7の場合は36文字×27行になります。

IchigoJamバージョン1.0.1と0.9.7とで画面の大きさが異なるため、LOCATE文を使用しているプログラムは異なるバージョンのIchigoJam上では正常に画面表示されないことがあります。

CLT文 (280行)とTICK()関数 (300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。

SCROLL文 (50行)

SCROLL文は画面表示を上下左右に1文字分だけシフト(スクロール)させます。上下左右のいずれにスクロールさせるかは、SCROLLの後に指定する数字(0:上/1:右/2:下/3:左)で選択します。

(例)SCROLL 0

上の例は画面表示を上方向に1文字分(1行分)スクロールさせます。

モールス符号（欧文）一覧表

本キットで取り上げた欧文のモールス符号の一覧表です。
（※和文のモールス符号もありますが、本キットでは扱っていないため割愛します）

文字	符号	文字	符号
A	・－	Z	－－・・
B	－・・・	.	・－・－・－
C	－・－・	:	－－－・・・
D	－・・	'	・－－－－・
E	・	(	－・－－・
F	・・－・	/	－・・－・
G	－－・	+	・－・－・
H	・・・・	,	－－・・－－
I	・・	?	・・－－・・
J	・－－－	－	－・・・・－
K	－・－	)	－・－－・－
L	・－・・	=	－・・・－
M	－－	″	・－・・－・
N	－・	@	・－－・－・
O	－－－	1	・－－－－
P	・－－・	2	・・－－－
Q	－－・－	3	・・・－－
R	・－・	4	・・・・－
S	・・・	5	・・・・・
T	－	6	－・・・・
U	・・－	7	－－・・・
V	・・・－	8	－－－・・
W	・－－	9	－－－－・
X	－・・－	0	－－－－－
Y	－・－－		

注：

(1) 左表の符号は、欧文(英文)用の標準的なモールス符号です。フランス語やドイツ語などヨーロッパ諸言語で使用される特殊な文字に対する符号は含まれていません。

(2) 訂正符号(文字を打ち間違えたときに打つ符号で、短点8個を連続して打つことになっています)は本キットでは使用していないため、省略しています。

(3) 「本文はじめ(BT)」、「本文おわり(AR)」など2文字分を連続して打つ符号は本キットでは使用していないため、省略しています。

モールス通信に使用される略号(主なもの)

モールス通信では通信を能率よく行えるよう、定型の文については略号が国際的に定められています。

覚えておくとモールス通信がより面白くなりますので、参考までに主なものを紹介します。

例文に登場する局名(コールサイン)は架空のものです。
（実在する局とは関係ありません）

◎ DE

「こちらは」の意味です。送信側の局名(コールサイン)を名乗るときに使用します。

（例）JCS JCS JCS DE GMK GMK GMK K

この例ではJCS局をGMK局が呼び出しています。

◎ K

送信する文の最後に置き、「受信します。応答どうぞ」の意味です。第三者に割り込まれては困る場合に使用する「KN」もあります。
使用例は上の「DE」の例文を見てください。

◎ CQ

受信している全ての局に対する呼び出し「どなたでも応答してください」の意味です。アマチュア無線の通信でもおなじみです。

（例）CQ CQ CQ DE JCS JCS JCS K

この例ではJCS局が「どなたでも応答してください」と呼びかけています。

◎ R

「了解しました」の意味です。本来は「QSL」を使用することになっているのですが、「R」のほうがよく使用されています。

（例）R GSS DE GMK

この例ではGMK局がGSS局に「了解しました」と応答しています。

◎ UR

「あなたの」の意味です。

◎ FR

英語の「FOR」の略です。

◎ TNX

「ありがとう」の意味です。アマチュア無線の交信証(QSLカード)によく「TNX FR UR QSL」と書いてあります。

◎ QSO

「QSO xx?」と打って「あなたはxx局と交信できますか?」、「QSO xx」と打って「私はxx局と交信できます」の意味です。

転じてアマチュア無線では「交信する」ことを「QSOする」と言っています。

◎ 73

国際的規則ではないのですが、交信の終わりに打つ「さようなら」の意味です。

同じく「さようなら」の意味の「VA」もあります。