今日(きょう)からプログラマー

ICHIGOJAM(いちごじゃむ) 初級(しょきゅう) 実機編(じっきへん)その03

サッカーPK戦

ギブテクウインウイン

「IchigoJam(R)jig.jp」

IchigoJam

目次（もくじ）

はじめに

今まで IchigoJam のモニターに画像を表示させてきました。

今度は IchigoJam マイコンボードの強みとなる外部出力端子を使って、サッカーのＰＫ戦ゲームを製作していきましょう。

あなたは　ゴールポストの左右いずれかを狙ってシュートを放ちます。

キーパーはマイコンなので強敵です。

連続でゴールしなければいけません。

キャッチされたらゲームオーバーです。最高得点を目指せ！

まず、準備された生基板に抵抗やＬＥＤを挿入しニッパでカットしていきます。

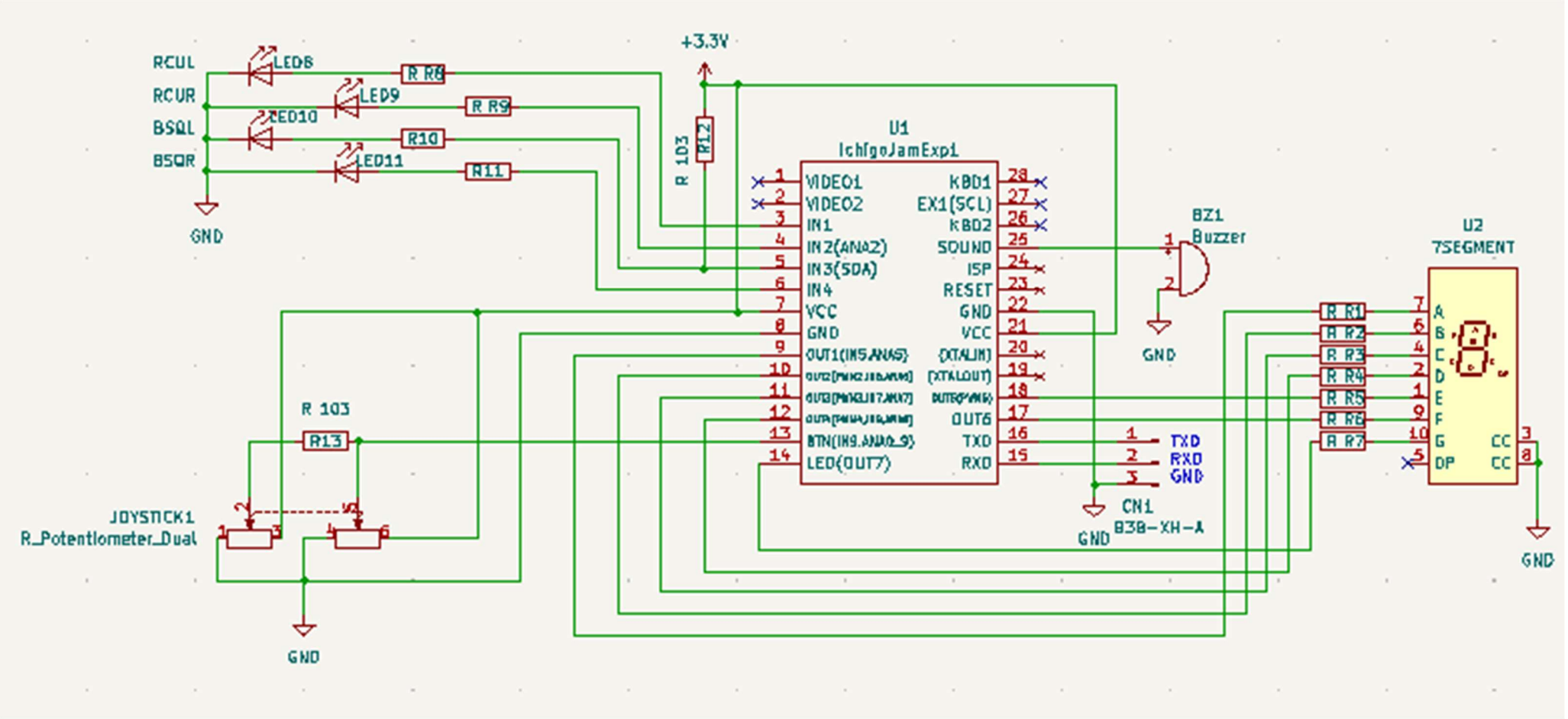
こんなヘンテコなサッカーを聞いたことがありますか？

それでは、レッツ　プレイ！

オレー　オレー

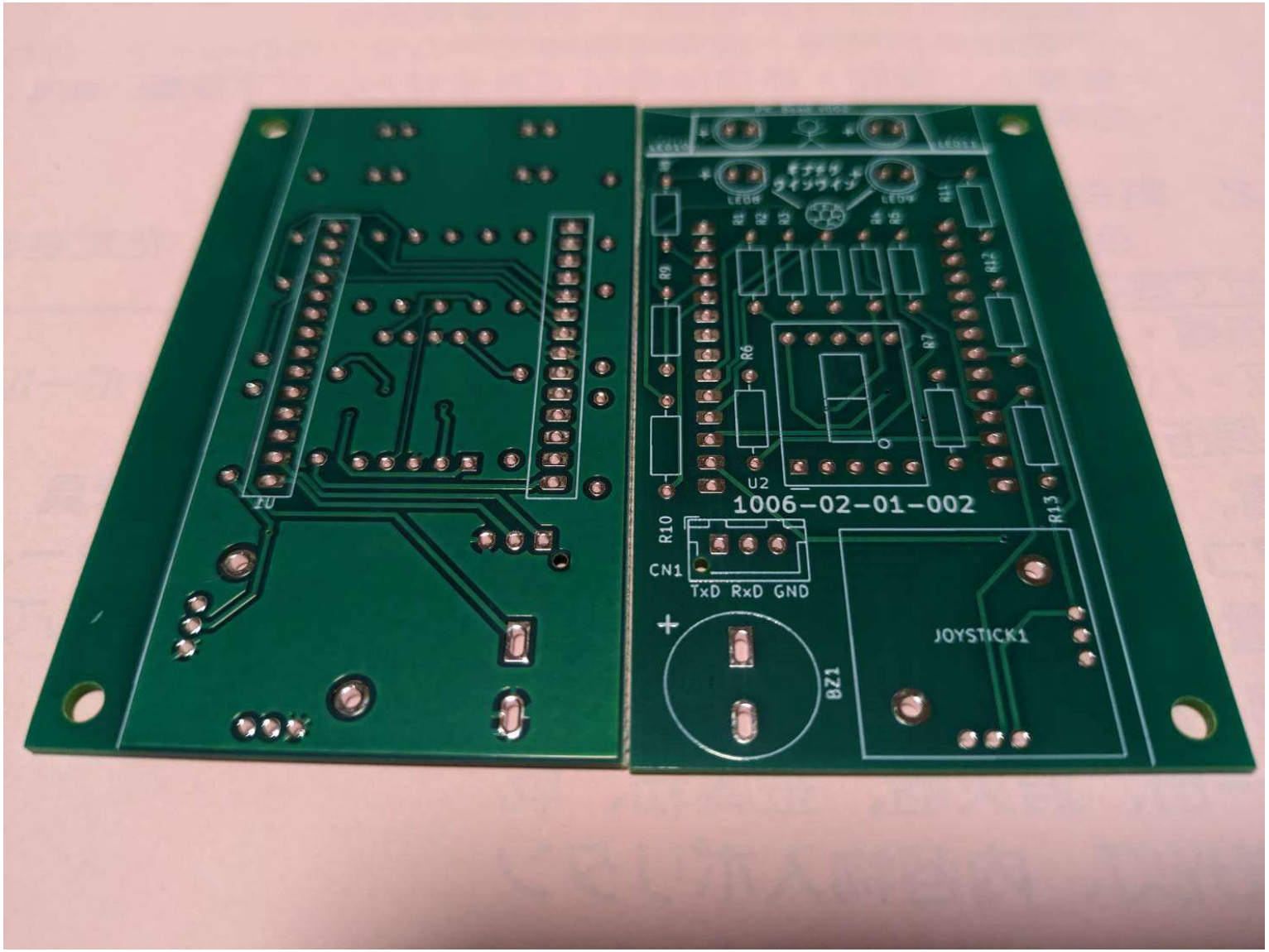
【01】電子工作　基板の回路図

製作する回路図



【02】製作する生基板のシルク図

裏面　　表面



【03】パーツリスト、マウントリスト

実装する部品と型式［］とシルク名称

カーボン抵抗　330Ω　1/4W　［CF25J330RB］11本　R1-11

カーボン抵抗　1kΩ　1/4W　［CF25J1KKB］2本　R12　R13

5mm　赤色LED　円筒　［φ5mm LED：赤色］2個　LED8-9

角型青色LED　［OSB5XA7DA4B-GH］2個　LED10-11

7セグエレメント表示器　カソードコモン　［OSL10561-LRA］1個　U2

アナログジョイスティック　［JT8P-3.2T-B10K-1-16Y］1個　JOYSTICK1

圧電スピーカー13mm　［PKM13EPYH4000-A0］1個 BZ1

コネクター　03P　［B3B-XH-A］1個 CN1

14PIN　［PSOKET-2.54-S-14P］2個　※裏面に実装すること

ピンヘッダー　14P　［127-1-50P］1個　U1　※14PINにカット

生基板　PK Boad V002［1006-02-01-002］1枚

【04】レイアウト図

キーパー青色 LED

ボール　赤色 LED

得点　7セグ LED

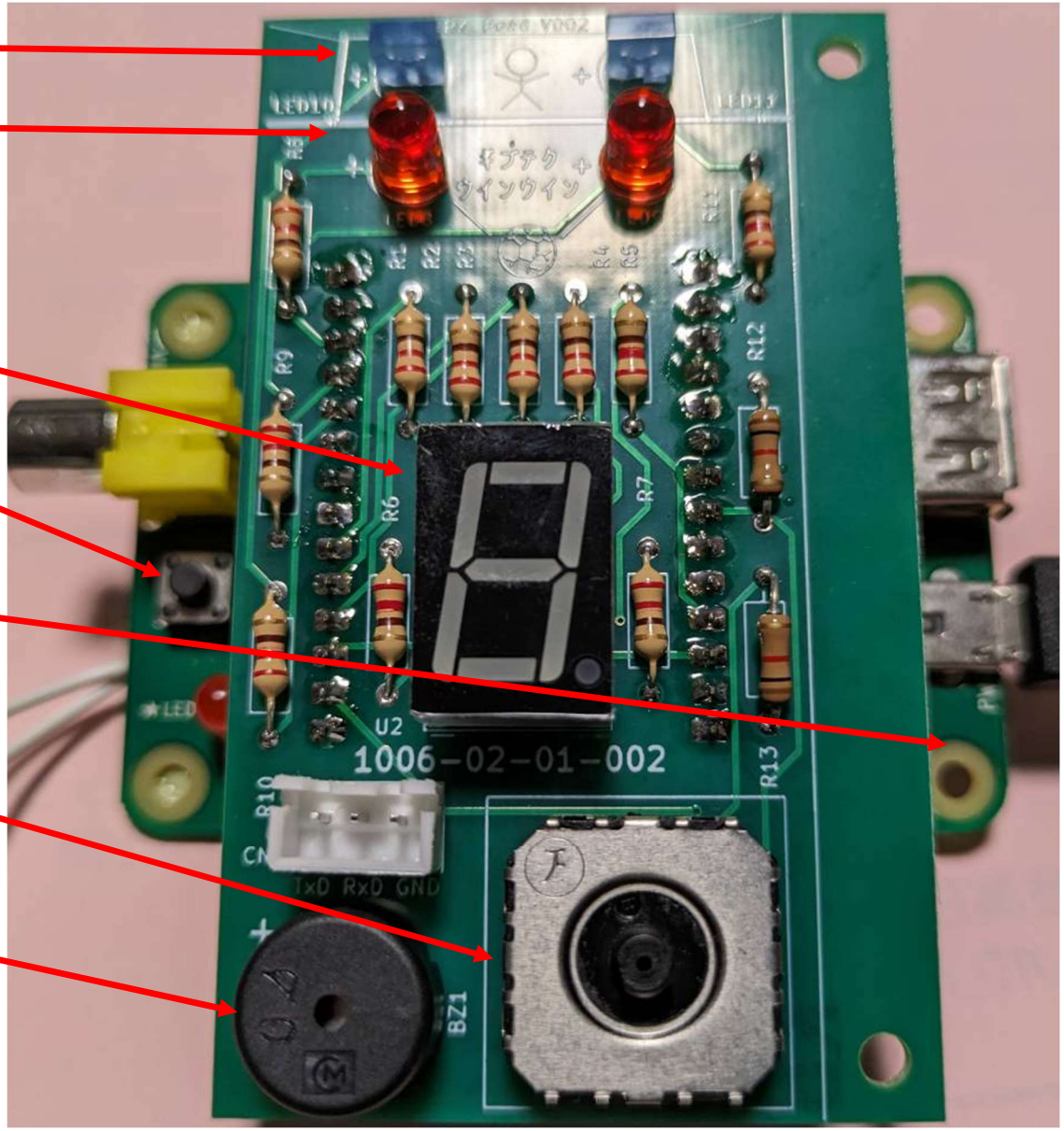
押しボタンSW

電源スライドSW

（下の IchigoJam 基板）

ジョイスティック

ブザー（なくてもよい）



【05】ゲームの概要

起動は電源スライドＳＷを左へスライドします。

すると、ボール LED（赤色）が左右交互に点滅します。

ボールをキックする方向を、右か左か選びます。

ジョイスティックを操作して選んだ方向にシュートすると、キーパーが跳びます。

キーパーの跳んだ方向の逆を突けばゴール成功で、続けてシュートできます。

ゴール成功で得点を示す７セグメントがインクリメント（＋１）されます。

キーパーに止められたらゲームオーバーで（0）に戻ります。

IchigoJam マイコンボードのプッシュボタンを押すとリプレイします。

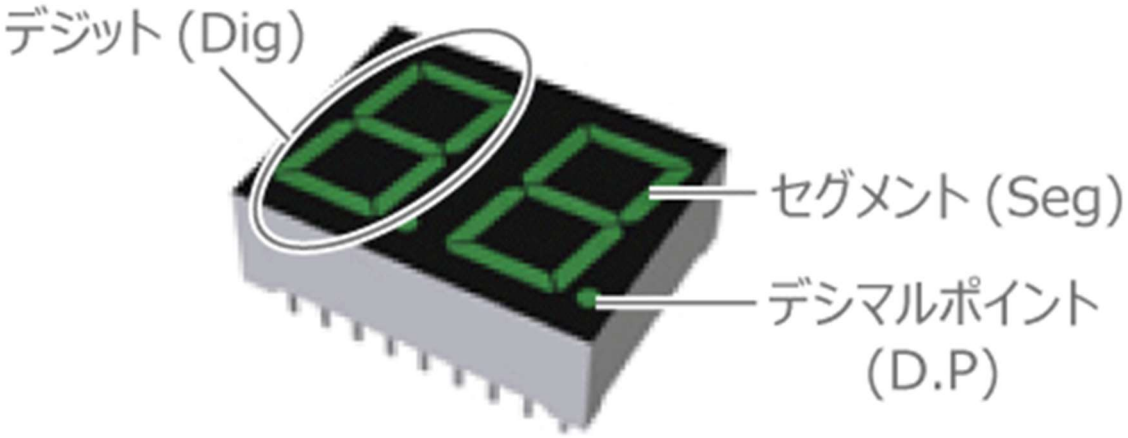
ジョイスティックの下方向でもリプレイします。

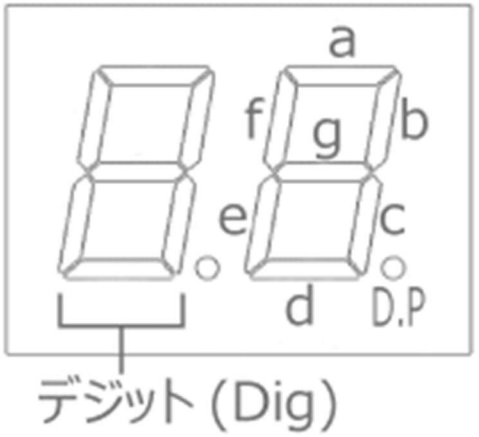
【06】7セグメントLED

7セグメントＬＥＤとは

7セグメントLEDは、数字情報の表示に特化したデジタル表示モジュールです。
表示する数字の形状部に発光ダイオード（LED）を配置しているため、大変視認性に優れています。
「LED数字表示器」や「7セグLED」と呼ばれることもあります。
7個のセグメントと1個の小数点となるデシマルポイントで構成されています。
今回製作するＰＫ戦の基板では　7セグメントは1桁のみであり、且つデシマルポイントはポートの関係上使用（接続）していません。

７セグメントＬＥＤメント LED 各部の名称





７セグメント LED の各部は次のように呼びます。

- 発光部（a～g）：セグメント (Seg)
- ドット発光部：デシマルポイント (D. P)
- a～g の７セグメントの総称：デジット (Dig)

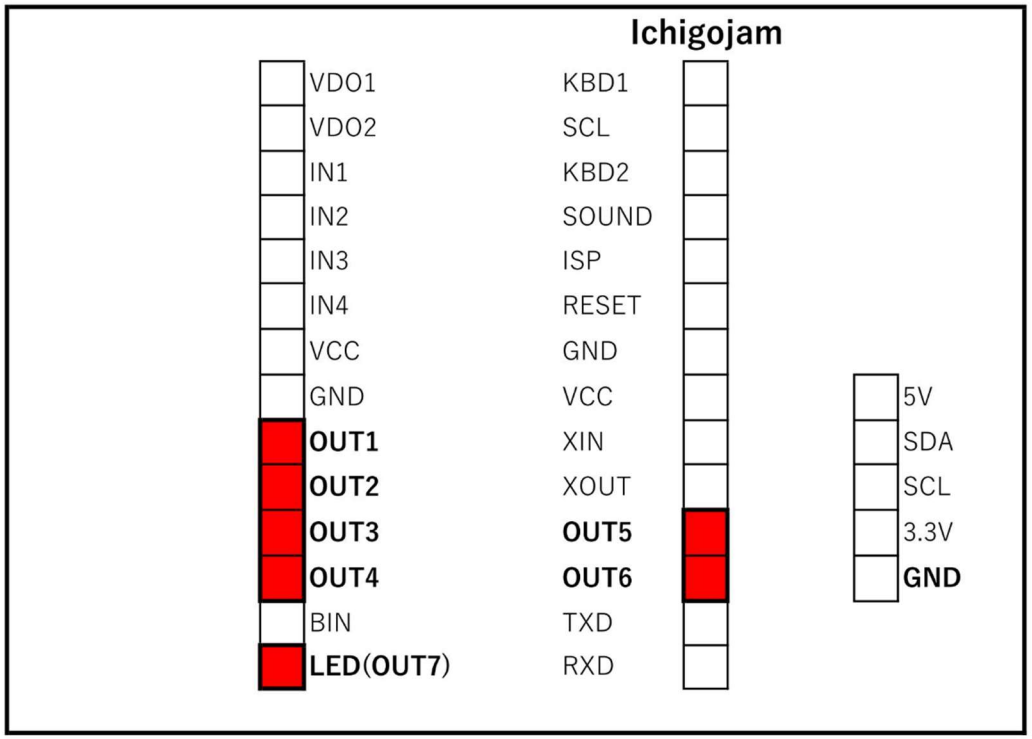
【07】0 から 9 までの数字をデータにする

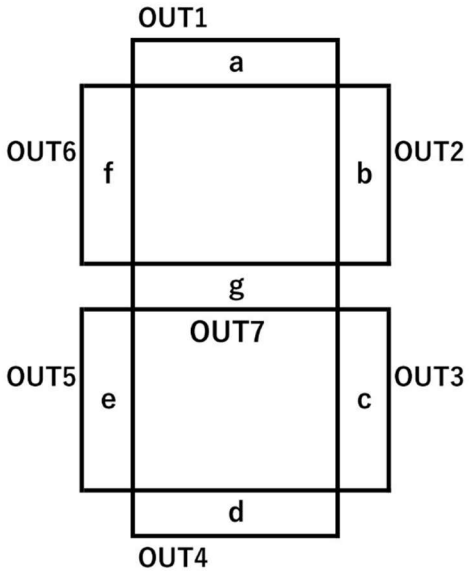
7 セグメントに 0 から 9 の数字を表示します。

この部分はハードウェアとの関連性が重要です。

ハードウェア毎(ごと)に表示のメカニズムが変わることに留意(りゅうい)する必要(ひつよう)があります。

今回の基板では　下記のように接続(せつぞく)されています。





OUT,

0	0	0	0	0	0	0	0
DP	g	f	e	d	c	b	a

OUT コマンド

OUT ［端子,］値

端子:端子番号（1～11）を指定します。OUT 7 端子は LED 端子と同じです。

値:端子を指定した場合は、値に 0、1、-1、-2 を指定します。

0 でローレベル、1 でハイレベルの信号を出力します。

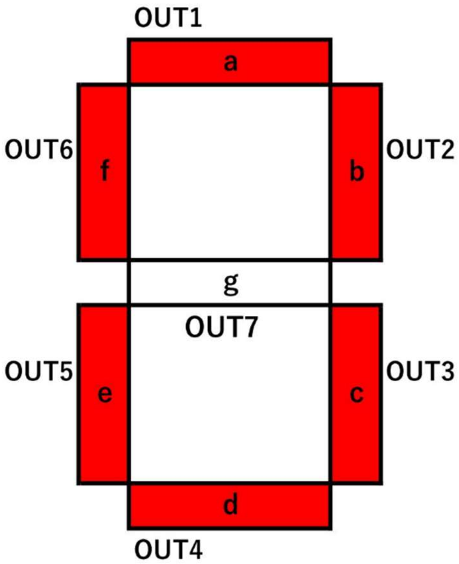
-1 を指定すると OUT1 から OUT6 を入力端子 IN5 から IN11 として利用できます。

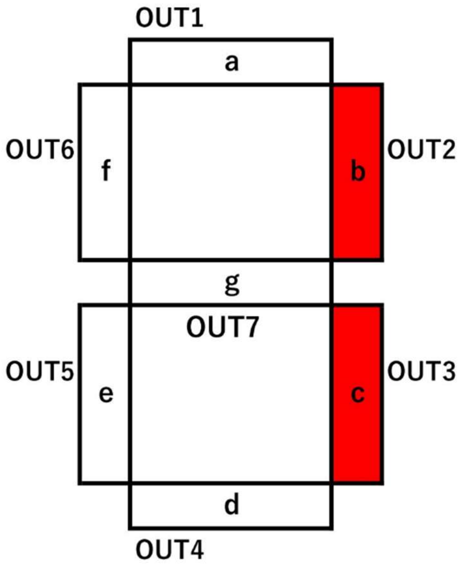
0 または 1 を指定して IN1 から IN4 を出力端子 OUT8 から OUT11 として利用できます。

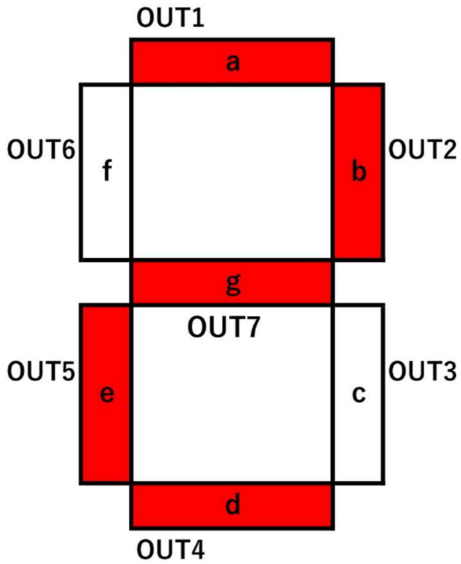
使用例

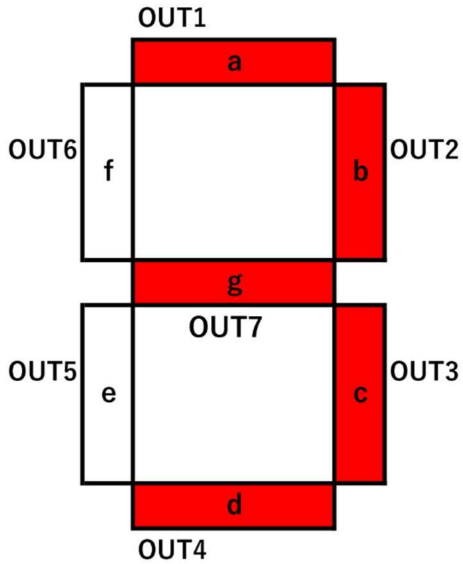
OUT 3, 1	OUT 3 端子にハイレベルを出力
OUT 12	OUT 3, OUT 4 端子をまとめてハイレベルを出力(12=`0001100)
OUT 1, -1	OUT 1 を IN 5 または ANA(5) 入力端子として利用できます。
OUT 2, -2	OUT 2 を IN 6 プルアップ付き入力端子として利用できます。
OUT 8, 0	IN 1 を OUT 8, 0 出力端子として利用できます。

0と1と2と3をデータ化する。









DP	g	f	e	d	c	b	a
0	0	1	1	1	1	1	1

DP	g	f	e	d	c	b	a
0	0	0	0	0	1	1	0

DP	g	f	e	d	c	b	a
0	1	0	1	1	0	1	1

DP	g	f	e	d	c	b	a
0	1	0	0	1	1	1	1

0

`00111111

#3F

1

`00000110

#06

2

`01011011

#5B

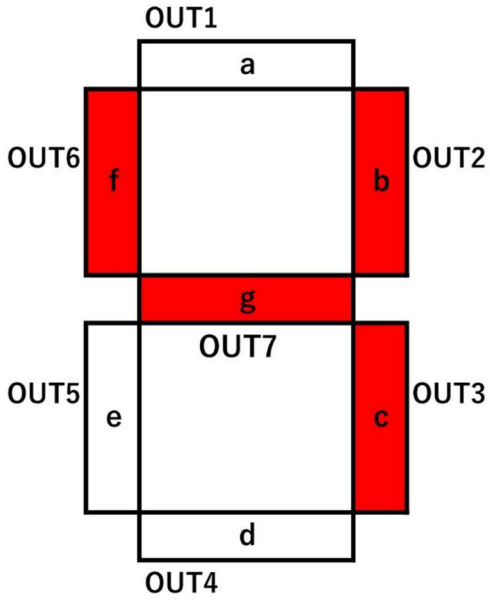
3

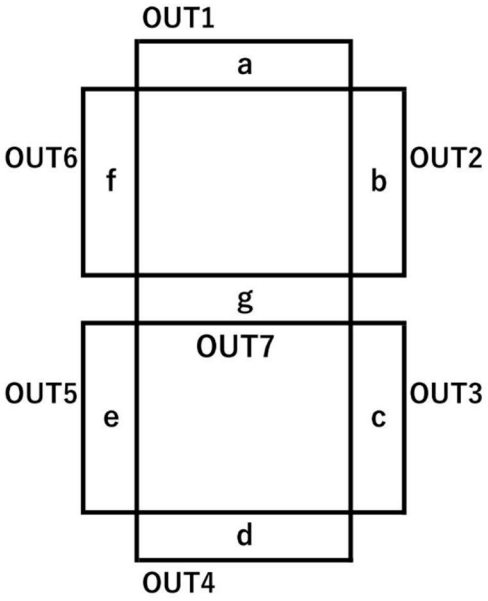
`01001111

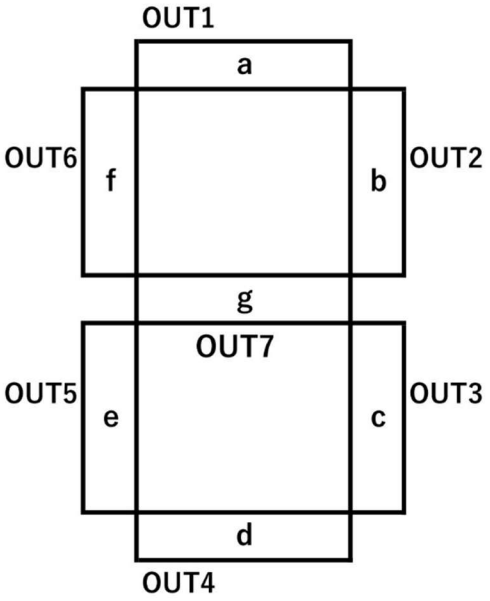
#4F

同様に4~9もデータ化してみよう。

4と5と6をデータ化する。







0							
DP	g	f	e	d	c	b	a

4

`

#

0	0	0	0	0	0	0	0
DP	g	f	e	d	c	b	a

5

`

#

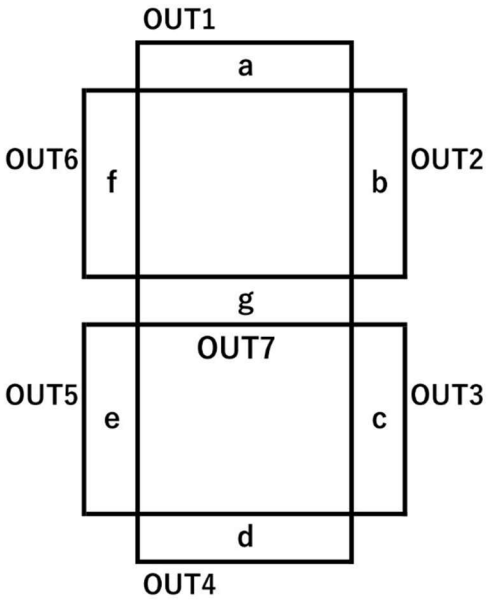
0	0	0	0	0	0	0	0
DP	g	f	e	d	c	b	a

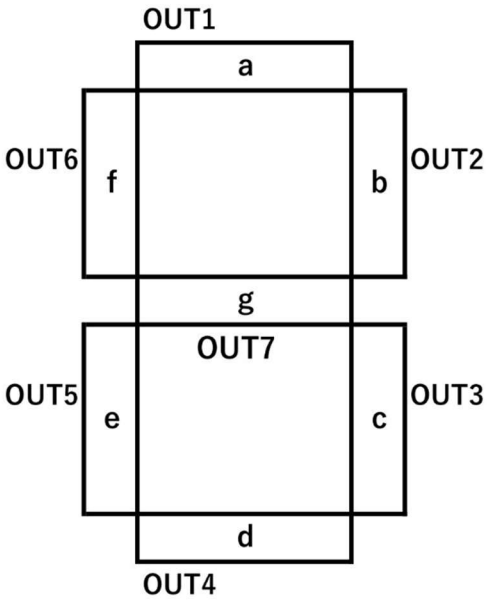
6

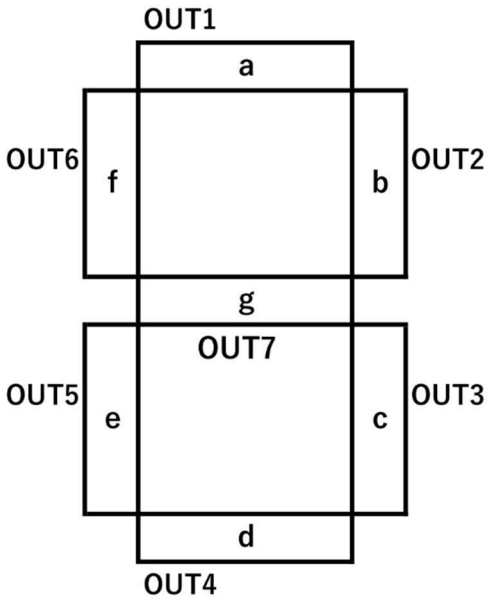
`

#

7と8と9をデータ化する。







0	0	0	0	0	0	0	0
DP	g	f	e	d	c	b	a

7

0	0	0	0	0	0	0	0
DP	g	f	e	d	c	b	a

8

0	0	0	0	0	0	0	0
DP	g	f	e	d	c	b	a

9

16 進数

2進数までは分かったが 16 進数への変換で行き詰まってしまうのではないでしょうか？

そこで２進数と関係が深い 16進数をみていきましょう。

コンピューターの処理は２進数で構成されています。

0 と 1 だけの数字が並んでいては　人が取り扱いにくいですね。そこで簡易に見るために 16進数に置き換えます。

16進数に置き換えても　難しいのですが、ここでは置き換え

方を学習していきます。

コンピューターの世界では、すべてのデータを 2進数で表しています。2進数というのは、1桁で 0 か 1 を使い表現しています。

2進数の 1桁では、0 か 1 なので 2通りになります。このコンピューターのデータ最小単位を **1ビット**といいます。

2進数の 2桁では、2 は`10、3 は`11、そしてまた桁上りします。

3桁なら 3 ビット、4桁なら 4 ビットとなっていきます。

`は2進数(しんすう)を示(しめ)します。（バック・クオート）

10進数(しんすう)の0から15まで

10進数	0	1	2	3	4	5	6	7
2進数	`0	`1	`10	`11	`100	`101	`110	`111

10進数	8	9	10	11	12	13	14	15
2進数	`1000	`1001	`1010	`1011	`1100	`1101	`1110	`1111

16進数表記

16進数は 0～9、A～F の 16種類の数字とアルファベットで表記します。

1桁で 16種類、16通り 表現できます。

10進数	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16進数	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

まとめましょう。

10進数(しんすう)　16進数(しんすう)　2進数(しんすう)　一覧表(いちらんひょう)

10進	0	1	2	3	4	5	6	7
16進	#00	#01	#02	#03	#04	#05	#06	#07
2進	`0000	`0001	`0010	`0011	`0100	`0101	`0110	`0111

10進	8	9	10	11	12	13	14	15
16進	#08	#09	#0A	#0B	#0C	#0D	#0E	#0F
2進	`1000	`1001	`1010	`1011	`1100	`1101	`1110	`1111

IchigoJam WEB で変換(へんかん)して、確(たし)かめよう。

16進数(しんすう)のゼロは#00　シャープを先頭(せんとう)に付けます。

2進数(しんすう)のゼロは`0000　`を先頭(せんとう)に付(つ)けます。

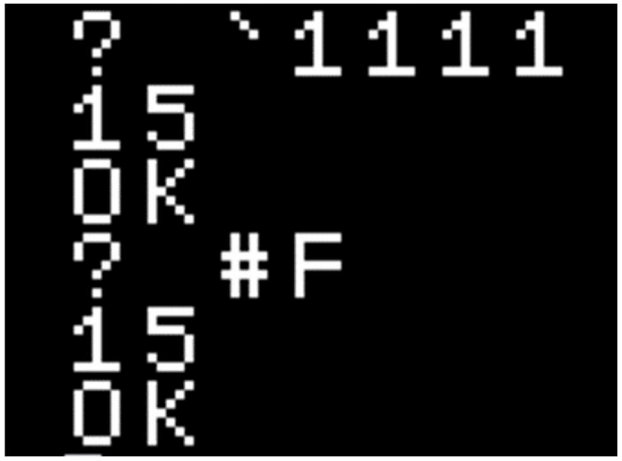
`は　[SHIFT]+ [@]のキーボードにあります。

? `1111

15

? #F

15



? `1111　と　? #F　はいずれも 10 進数で 15 になります。

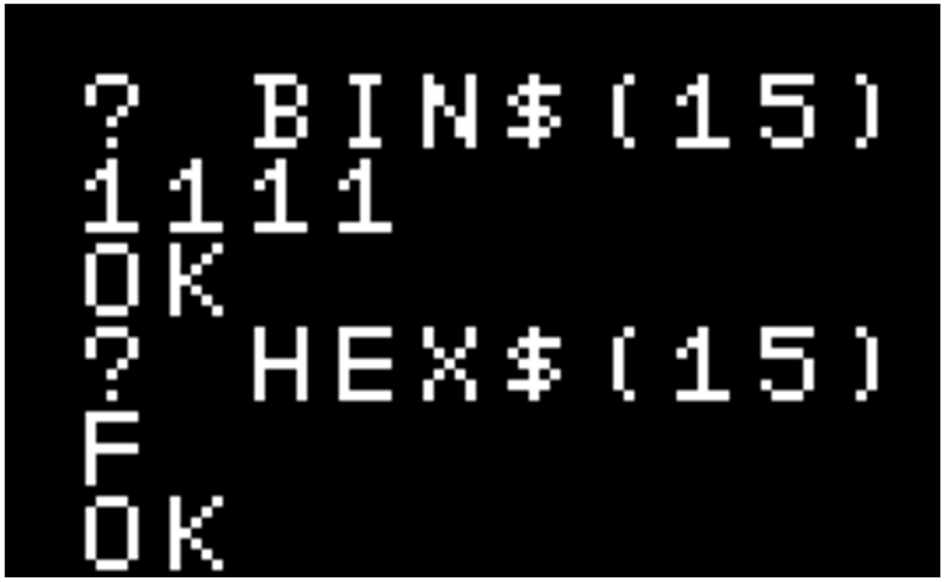
今度(こんど)は 10進数(しんすう)を　２進数(しんすう)や　16進数(しんすう)に変換(へんかん)するコマンドを
実験(じっけん)してみましょう。ＢＩＮ＄（）とＨＥＸ＄（）です。
10進数(しんすう)の 15 を変換(へんかん)してみましょう。

```
 ? BIN$(15)

1111

 ? HEX$(15)

F
```



`1111 と　#F になります。

10進数の　0 から 15

16進数の　#00 から#0F

2進数の　`0000 から　`1111

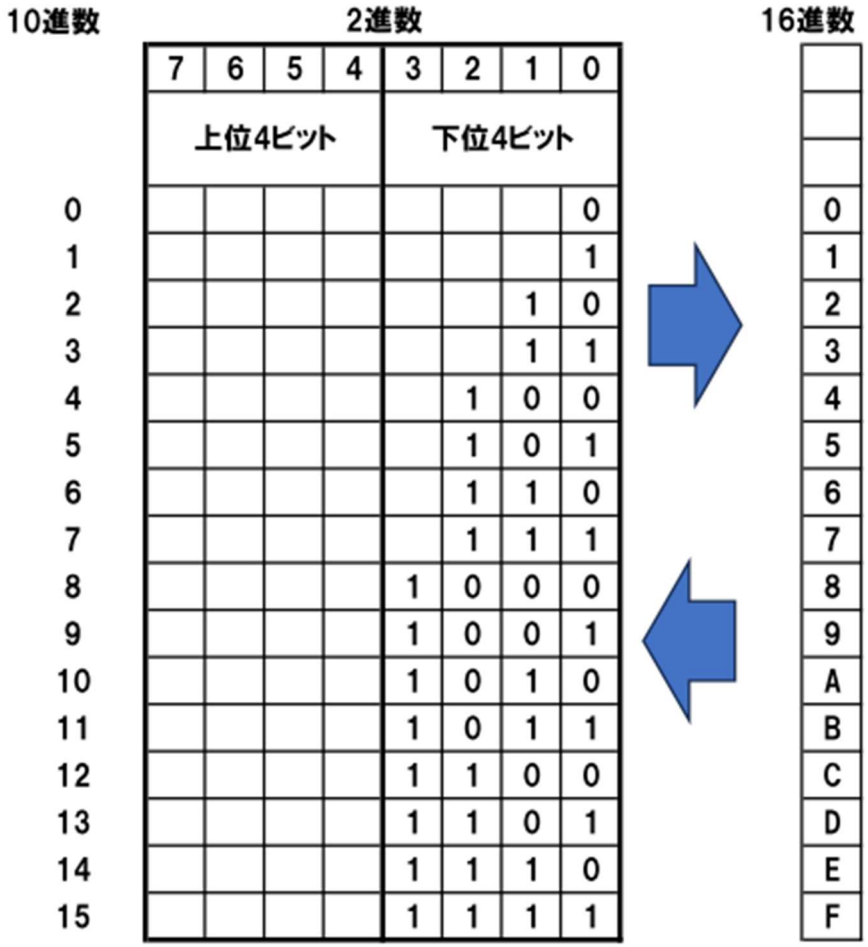
を勉強してきました。

ここまでが　重要な基礎になります。

2進数でいう 4桁は　4bit となり　あとは組み合わせて表現していくことができます。

今回は 1 バイト（8ビット）までを取り扱います。

2進数の 4桁が、ちょうど 16進数の 1桁にピッタリ収まっているのが重要です。



＃0から＃Ｆの次

＃10から＃1Ｆも観察(かんさつ)しましょう。

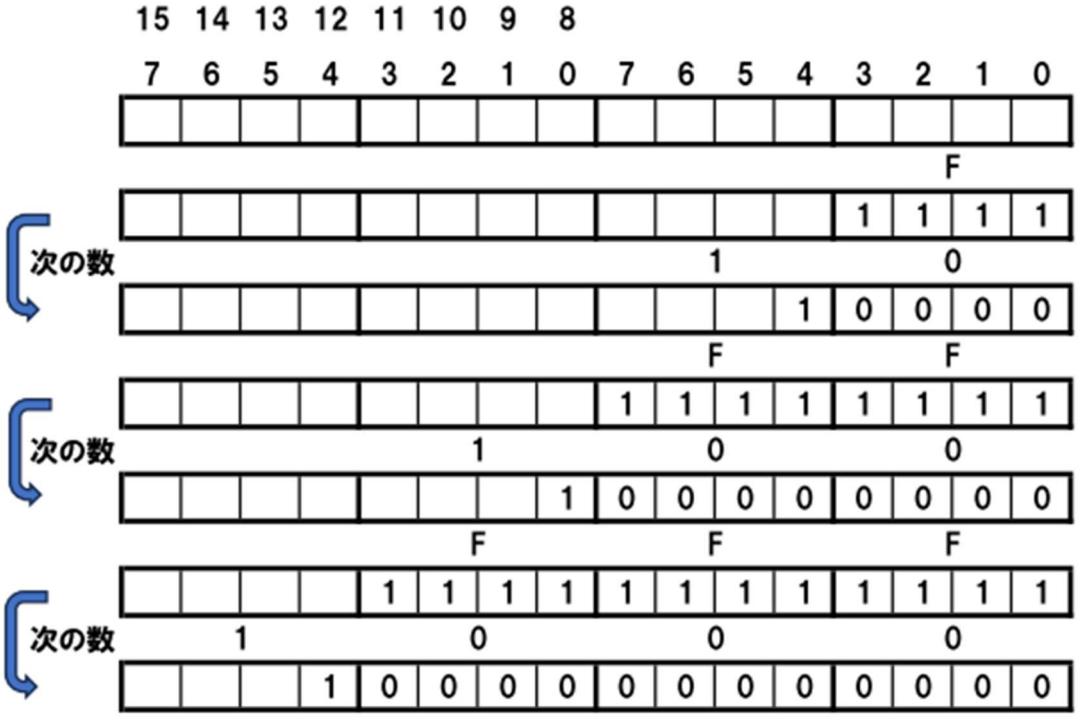
何(なに)か気(き)が付(づ)きませんか？

下位(かい)ビットの網線部(あみせんぶ)が

繰(く)り返(かえ)されていますよね。

10進数	2進数								16進数
	7	6	5	4	3	2	1	0	
	上位4ビット				下位4ビット				
0					0	0	0	0	0
1					0	0	0	1	1
2					0	0	1	0	2
3					0	0	1	1	3
4					0	1	0	0	4
5					0	1	0	1	5
6					0	1	1	0	6
7					0	1	1	1	7
8					1	0	0	0	8
9					1	0	0	1	9
10					1	0	1	0	A
11					1	0	1	1	B
12					1	1	0	0	C
13					1	1	0	1	D
14					1	1	1	0	E
15					1	1	1	1	F
16				1	0	0	0	0	10
17				1	0	0	0	1	11
18				1	0	0	1	0	12
19				1	0	0	1	1	13
20				1	0	1	0	0	14
21				1	0	1	0	1	15
22				1	0	1	1	0	16
23				1	0	1	1	1	17
24				1	1	0	0	0	18
25				1	1	0	0	1	19
26				1	1	0	1	0	1A
27				1	1	0	1	1	1B
28				1	1	1	0	0	1C
29				1	1	1	0	1	1D
30				1	1	1	1	0	1E
31				1	1	1	1	1	1F

この図（ず）は＃Ｆから＃10、＃ＦＦから＃100など桁上（けたあ）がりの様子を表（あらわ）しています。



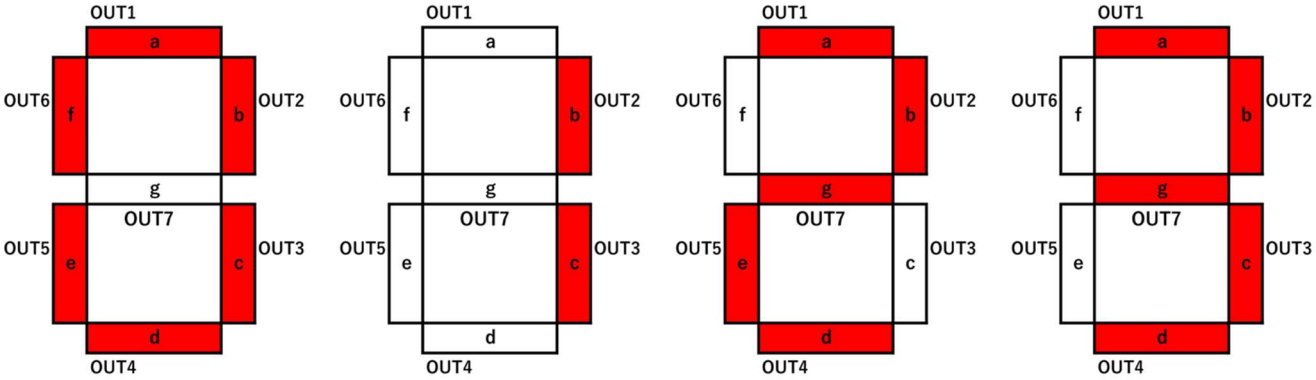
1バイト（8ビット）の＃0から＃ＦＦの場合

上位4 ビットと　下位4 ビットを完全に分離して考えることができます。

例えば左端0 の場合

`00111111 は

0011　と 1111 とし



＃3　F　と上位4 ビット　下位ビットにわけて表記できます。

コンピューターの世界では、すべてのデータを 2進数で 表 しています。2進数というのは、1桁で 0 か 1 を使い 表 現しています。

　マイコンの世界では　4bit　8bit　16bit と倍々に拡大されてきました。

4bit までの 表 を書く事が出来るように練 習 しましょう。
忘れた場合でも　導 けるようになることが大切です。

今(いま)まで　大変(たいへん)むずかしい　勉強(べんきょう)をしてきました。

全(まった)くわかりません。という方(かた)が大半(たいはん)かもしれません。

しかし　あきらめないでください。

ここは　必要(ひつよう)な時(とき)がくれば　読(よ)み返(かえ)せばいいのです。

わからないところは飛(と)ばしても問題(もんだい)ありません。

楽(たの)しい部分(ぶぶん)だけ　勉強(べんきょう)していき

わからないときや　必要(ひつよう)なときは

テキストを読(よ)み返(かえ)せばいいのです。

【08】フローチャートとプログラム

ジョイスティックはBTN端子につながっており、アナログ入力として使えます。
「ANA()」関数で、左端:128以下、右端:896以上として読み取れます。

G：ゴール得点変数

D：ボール変数　0が左　1が右（キッカーの選手がゴールにボールを蹴った方向）

A：ジョイスティック入力変数　左の時は-1　右の時は1　中間の時は0

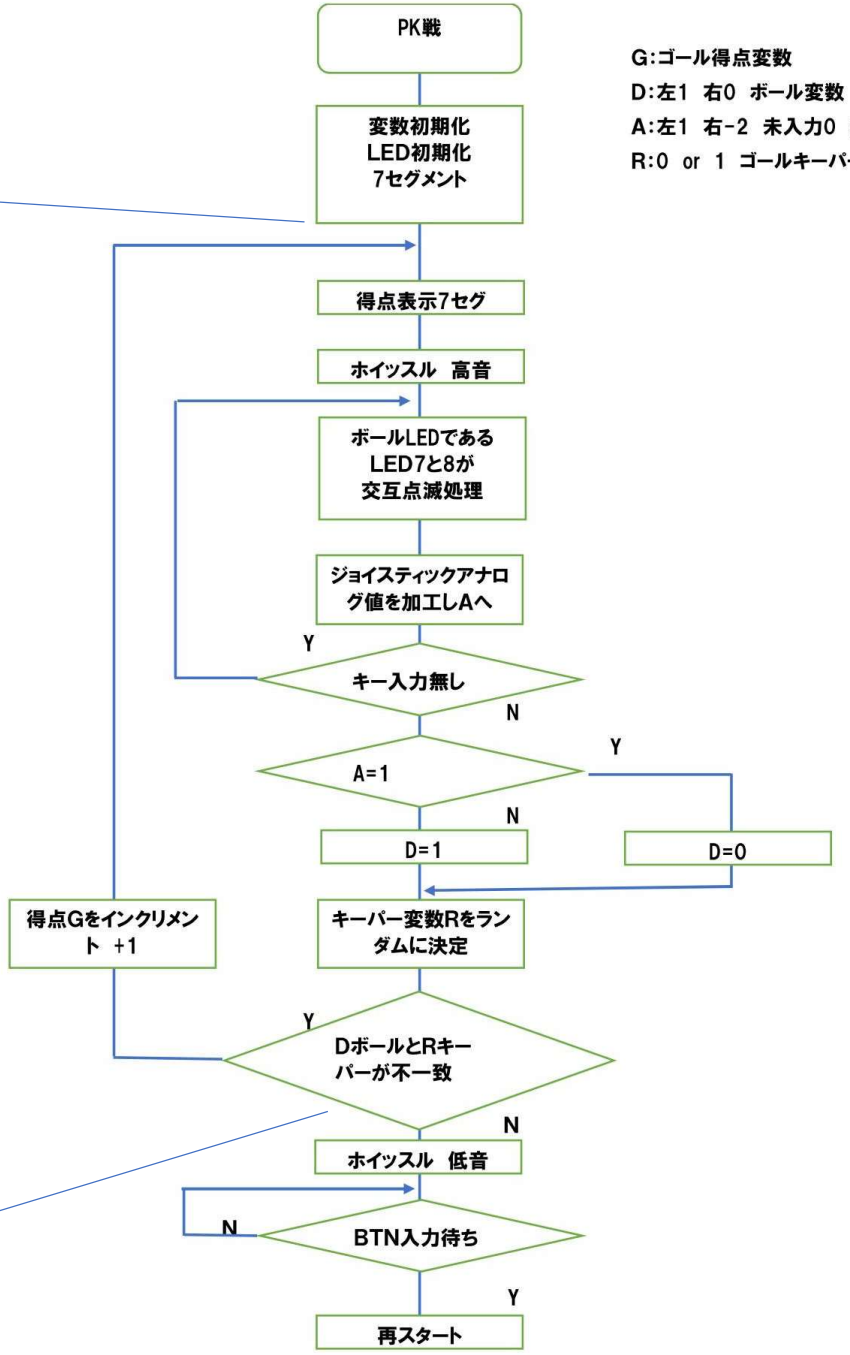
R：ゴールキーパー変数　0が左　or　1が右（ゴールキーパーがボールを取りに行った方向）

サッカーＰＫ戦プログラム

```
10 @ARUN:'*PK
20 CLV
30 FOR P=1 TO 11:OUT P,0:NEXT
40 LET[0],`0111111,`0000110,`1011011,`1001111,`1100110,`1101101,`1111101,
```

行つづく

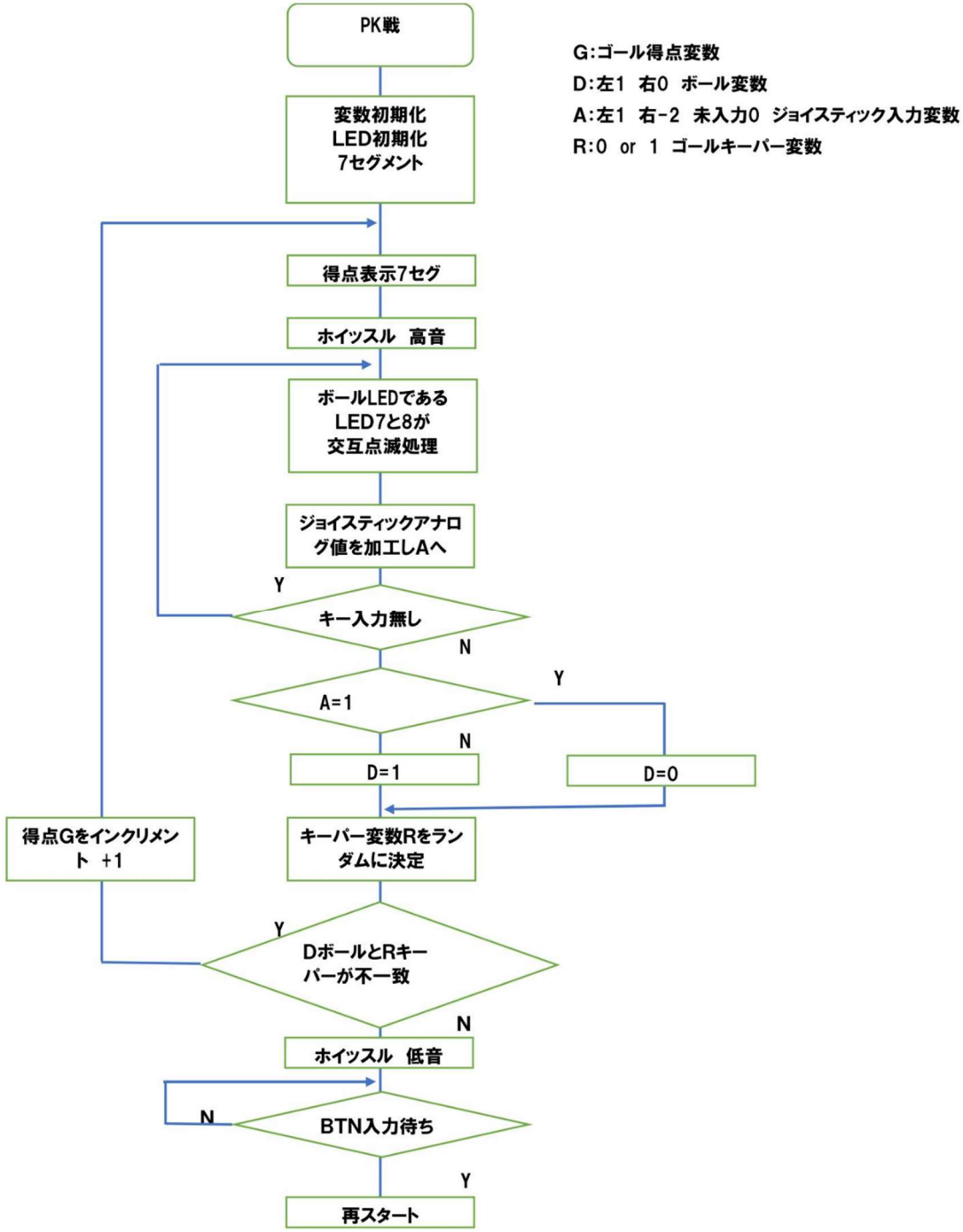
```
`0100111, `1111111, `1101111
50 @START
60 OUT [G%10]
70 BEEP 10,30
80 WAIT 30
90 @IND
100 D=1-D
110 OUT 8+D,1
120 WAIT 3
130 OUT 8+D,0
140 WAIT 3
150 A=(ANA()-512)/256
160 IF A=0 GOTO @IND
170 D=(A+1)/2
```



G:ゴール得点変数
D:左1　右0　ボール変数
A:左1　右-2　未入力0　ジョイスティック入力変数
R:0　or　1　ゴールキーパー変数

```
180 BEEP
190 OUT 8+D, 1
200 WAIT 30
210 SRND TICK()
220 R=RND(2)
230 OUT 10+R, 1
240 WAIT 30
250 IF D!=R G=G+1:GOTO @START
260 BEEP 30, 60
270 WAIT 60
280 IF !BTN() CONT
290 RUN
```

フローチャート


PK戦
変数初期化
LED初期化
7セグメント
G:ゴール得点変数
D:左1 右0 ボール変数
A:左1 右-2 未入力0 ジョイスティック入力変数
R:0 or 1 ゴールキーパー変数
得点表示7セグ
ホイッスル 高音
ボールLEDである
LED7と8が
交互点滅処理
ジョイスティックアナロ
グ値を加工しAへ
Y
キー入力無し
N
Y
A=1
N
D=1
D=0
得点Gをインクリメン
ト +1
キーパー変数Rをラン
ダムに決定
Y
Dボールと Rキー
パーが不一致
N
ホイッスル 低音
N
BTN入力待ち
Y
再スタート

【09】解説

10 @ARUN:'*PK

→ 電源(でんげん)を入れた時に save0 したプログラムを自動(じどう)スタートさせたい時に @ARUN と書きます。

→ ' または REM : プログラムを作成(さくせい)していると、この行は何をしているのかを示(しめ)しておきたい場合(ばあい)があります。このような場合にプログラム中にコメント（注釈(ちゅうしゃく)）を入れておくことができます。REM と入力し、その後(あと)にコメントを書きます。REM の次の文字から、その行の最後(さいご)までがコメントになります。

20 CLV

→ 変数(へんすう)と配列(はいれつ)変数を全(すべ)て 0 にします。（変数を 0 に初期化(しょきか)すると言います。）

30 FOR P=1 TO 11:OUT P,0:NEXT

→ FOR _ TO _ (STEP:省略可能(しょうりゃくかのう)) 文 NEXT

P=1 から始(はじ)めて、P=11 になるまで NEXT までの文を繰(くり)り返(かえ)します。

STEP省略時はSTEP1となりPの値が1つずつふえていきます。

つまり　OUT 1,0 から　OUT 11,0 となり　7セグメントとLED全てを消灯します。

また、IN1はOUT8、IN2はOUT9、IN3はOUT10、IN4はOUT11に設定されます。

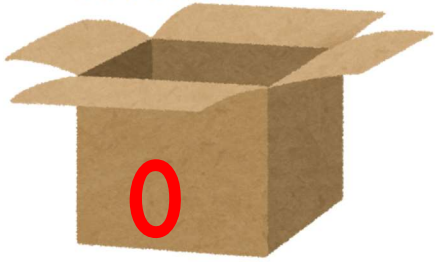
```
40 LET[0], `0111111, `0000110, `1011011, `1001111, `1100110, `1101101, `1111101, `0100111,
`1111111, `1101111
```

→　[]は、配列といい、LET [数], 数,, 数nは、配列の値を数で設定します。

[0]に`0111111、[1]に`0000110と順番に2進数の値を入れていき

最後に[9]に`1101111を入れます。

LET[0]　`0111111



LET[1]　`0000110

中略

LET[9]　`1101111

50 @START

→　行の先頭に＠を書くとラベルになります。行番号の代わりとして使います。
（GOTO @START など）

GOTO の後に　直接行番号を指定してもよいが　プログラムの行を増やしたり
すると行番号が　ずれてしまいジャンプしたい行番号を修正する必要が発生し
ます。このためラベルを使うことで　間違いにくくしたり　ほかの人が見て
もわかり易くします。

60 OUT [G%10]

→　得点変数 G の値を 10 で割った余りの値の配列を 7 セグメントに出力します。
つまり G の値の 1 桁目を数字として 7 セグメントに表示します。

70 BEEP 10,30

→　BEEP　周期（1～32767），長さ（1/60 秒単位）　両方とも省略可能です。
周期：値が小さいほど高い音、大きいと低い音になります。
長さ:60 を指定すると約 1 秒間音が鳴ります。

80 WAIT 30

→　WAIT 長さ（1/60 秒単位）

　　WAIT 60 で 1 秒待つことになります。　2 秒は WAIT 120　, 0.5 秒は WAIT 30

90 @IND

→　ラベル　50 行の説明参照

100 D=1-D

→　初めて出てくる変数ですが　20 行の CLV で最初は 0 に初期化されています。

　　ボール変数 D は 0: 左になったり 1: 右になったりします。

　　計算結果　D には 1: 右または、0: 左が代入されます。左と右に交互に変化

110 OUT 8+D, 1

→　D の値により出力端子OUT8 または、OUT9 に 1 を出力して LED を点灯します。

　　100 行の　計算式によってボール変数が 0 になったり 1 になったりするので

　　出力端子につながっている LED8:左が点灯したり、LED9:右が点灯したりします。

120 WAIT 3

→　3/60＝0.05 秒待ちます。

130 OUT 8+D,0

→　D の値によって 110 行で点灯した LED8:左　または、LED9:右を消灯(しょうとう)します。

140 WAIT 3

→　3/60＝0.05 秒待ちます。

150 A=(ANA()-512)/256

→　BTN 端子はアナログ入力として使え「ANA()」関数(かんすう)で、ジョイスティックの位置(いち)により左端(ひだりはし):128 以下、右端(みぎはし):896 以上と読み取れます。
計算式(けいさんしき)よりジョイスティック入力(にゅうりょく)変数 A の値は　左:A=-1、右:A=1、中間(ちゅうかん):A=0

160 IF A=0 GOTO @IND

→　ジョイスティックが左か右に、たおされていないときは　@IND へジャンプ
　　ジョイスティックが 0 以外では、左や右の時は次の行へ

170 D=(A+1)/2

→　ジョイスティックが左:A=-1 の時　ボール変数　左:D=0　とします。

　　ジョイスティックが右:A=1　の時　ボール変数　右:D=1　とします。

180 BEEP

→　短(みじか)いビープ音

190 OUT 8+D,1

→　ボール変数　左:D=0　の時　LED 8 ON（左ボール点灯）左にキック

　　ボール変数　右:D=1　の時　LED 9 ON（右ボール点灯）右にキック

200 WAIT 30

→　30/60＝0.5 秒待ちます。

210 SRND TICK()

→　乱数(らんすう)の種(たね)にタイマー(1/60 秒で 1 進む)を使用します。

220 R=RND(2)

→　ゴールキーパー変数　R に 0:左か 1:右が乱数として代入される。

```
230 OUT10+R,1
```

→　ゴールキーパー変数　R により OUT10 または、OUT11 に 1 を出力して LED を点灯します。ゴールキーパーが　左:R=0、右:R=1 に跳(と)んでボールを取ります。

```
240 WAIT 30          →　0.5 秒待つ
250 IF D!=R G=G+1:GOTO @START
```

→　ボール変数　D とゴールキーパー変数　R　の値が違(ちが)う時は、得点変数(とくてんへんすう) G を＋1して@START へジャンプし、繰(く)り返(かえ)しシュートできます。
キッカーのキックしたボールの方向(ほうこう)とキーパーの方向が違う時はゴールです。

```
260 BEEP 30,60    →　ビープ　低(ひく)い音 1 秒間
270 WAIT 60          →　1 秒待つ
280 IF !BTN() CONT
```

→　CONT　は、その行番号を繰(く)り返(かえ)します。押(お)しボタン BTN が押されたら次の行へ

```
290 RUN
```

→　最初から再スタート

【10】0から9までを1秒間隔で表示してみよう

各自TRY

【11】aからzまでのデータを制作し、自分の名前を1秒間隔で表示してみよう

各自TRY

ぎじゅつしりょう
技術資料

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

コマンド解説資料　※ルビは省略します

赤文字は省略可能　青文字は説明文

●式/演算

［加算］数＋数

［減算］数－数

［乗算］数＊数

［徐算］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

●代入／変数／配列変数

［数］　0～101 まで

ＬＥＴ　変数，数　省略形：変数＝数

ＬＥＴ　［数］，数，…数ｎ

●リセット／初期化

ＣＬＫ　キーバッファ消去

ＣＬＰ　パターン初期化

ＣＬＳ　画面消去

ＣＬＴ　時間をリセット

ＣＬＶ　変数を０に消去

●キー入力／ボタン

ＢＴＮ（0/UP/DOWN/RIGHT/LEFT/SPACE）

ＩＮＫＥＹ（）　リアルタイムキー入力

ＩＮＰＵＴ　文字列，変数

●画面関係

ＬＯＣＡＴＥ　Ｘ座標，Ｙ座標　省略形：ＬＣ

ＰＲＩＮＴ　数や文字　省略形：？

ＳＣＲ（Ｘ座標，Ｙ座標）

ＳＣＲＯＬＬ　数　０：上、１：右、２：下、３：左

ＶＩＤＥＯ　数１，数２　ＶＩＤＥＯ　０で画面表示されなくなり　ＶＩＤＥＯ　１で画面が表示される。

ＶＩＤＥＯ　３で拡大モード

●関数

ＡＢＳ（数）　絶対値（ぜったいち）

ＡＳＣ（“文字”）　アスキー文字コード

ＢＩＮ＄（数，桁数）　２進数の文字列

ＣＨＲ＄（数，…数ｎ）　文字コードに対する文字列

ＤＥＣ＄（数，桁数）　10進数の文字列

ＨＥＸ＄（数，桁数）　16進数の文字列

ＲＮＤ（数）　0から数未満のでたらめな整数

●数値表現

123　10進数（-32768～32767）

#E9　16進数（0～#FFFF）

‘1011　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　　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

　文　～ＮＥＸＴ

　変数を数１から数２までを数３きざみで NEXT までの

　文を繰り返し実行する。

ＧＯＳＵＢ　行番号　省略形：ＧＳＢ　行番号から始まる

　サブルーティンを実行する。

ＧＯＴＯ　行番号　行番号にジャンプする。

ＲＥＴＵＲＮ　省略形：ＲＴＮ　サブルーティンを呼んだ

ところに戻る。

●ハードウェア

ＡＮＡ（数）　0～1023　アナログ入力 0～3.3V（電圧値）

ＢＰＳ　通信速度　省略時：115200bps

ＩＣＲ（数1，数2，数3，数4，数5）　I2C 読み込み

ＩＣＷ（数1，数2，数3，数4，数5）　I2C 書き込み

ＩＮ（数）　入力端子　IN1-11 から入力

ＢＴＮ　入力端子　（IN 9 端子、ANA 0 端子、ANA 9 端子）

ＬＥＤ　数　0：消灯　1：点灯　（OUT 7 端子）

OUT　数1，数2　　OUT1-11に出力する。

PWM　数1，数2，数3　OUT2-5にパルス出力する。

RESET　リセット

SLEEP　スリープ　BTNボタンを押すと復帰する。

UART　シリアル出力設定，シリアル受信設定

WAIT　数　60で1秒休止する。

●ファイル

FILES　数1，数2　数1から数2のsaveされた
　プログラムリストを表示する。0は全てを表示する。

LOAD　数　save数に保存されたプログラムを読み込む。

LRUN　数，行番号　save数に保存されたプログラム
　を読み込み行番号から実行する。

RUN　プログラムを最初から実行する。

SAVE　数　プログラムをsave数　領域に保存する。

●プログラム

ＣＯＮＴ　再度実行する。

ＥＮＤ　プログラムを終了する。

ＦＲＥＥ（）プログラムの残りメモリ数を返す。

ＬＩＮＥ（）現在進行中の行番号を返す。

ＬＩＳＴ　行番号１，行番号２　プログラムを行番号１から
　行番号２まで表示する。

ＮＥＷ　プログラムを消す。

ＲＥＮＵＭ　数１，数２　プログラムの行番号を数１から

数２の間隔で付け直す。

ＳＴＯＰ　処理を中断する

●メモリ操作／マシン語

ＰＥＥＫ（アドレス）　メモリーのアドレスにあるデータを読み出す。

ＰＯＫＥ　アドレス，数，…数ｎ　メモリーのアドレスに数を書き込む。

ＵＳＲ（アドレス，数）　メモリーのアドレスにあるマシン語を数を渡して実行する。

●その他

ＨＥＬＰ　メモリマップを表示

ＲＥＭ　　注釈　省略形：‘

ＴＩＣＫ（）　ＣＬＴからの時間（1/60）を返す

ＶＥＲ（）　バージョン番号を返す。

●音楽／サウンド

ＢＥＥＰ　周期，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（Ｃ Ｄ Ｅ FG Ａ Ｂ Ｒ）

音ｎ　音長（.をつけると 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　デフォルトの音調　（初期値：4）

Ｏｎ　オクターブ（1～5　初期値：4）

＞　　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（1～255）

10進数　16進数　2進数　一覧表

10進	0	1	2	3	4	5	6	7
16進	#00	#01	#02	#03	#04	#05	#06	#07
2進	`0000	`0001	`0010	`0011	`0100	`0101	`0110	`0111

10進	8	9	10	11	12	13	14	15
16進	#08	#09	#0A	#0B	#0C	#0D	#0E	#0F
2進	`1000	`1001	`1010	`1011	`1100	`1101	`1110	`1111

文字コード表　16 進数

```
10 ' *CHR16*
20 CLS
30 ?"     0123456789ABCDEF"
40 FOR J=0 TO 15
50 ?"#";HEX$(J*16,2)
60 FOR I=0 TO 15
70 A=#900+J*32+I+36
80 C=J*16+I
90 POKE A,C
100 NEXT
110 NEXT
```

文字コード表　10 進数

```
10 '*CHR10*
20 CLS
30 PRINT "                    1111111111"
40 PRINT "...+0123456789012345678 9"
50 FOR J=0 TO 12
60 PRINT J*20
70 FOR I=0 TO 19
80 A=#900+J*32+I+68
90 C=J*20+I
100 IF C<256 THEN POKE A,C
110 NEXT
120 NEXT
```

例)

左下の飛行機は？CHR$(240)：　右下のイチゴは?CHR$(255)：