

第5章 コマンド・リファレンス

5.1 変数と定数

TNYFSC P版では - 8323072 ~ 8323072 の整数を扱うことができます。(Z版では ± 32767)
扱える整数は、定数・変数・配列・関数の4種類です。

1) 定数

定数は、10進と16進が用意されており、それぞれ次の様に表現できます。

```
1234    &H123
```

実際には、次の様に使用されます。

```
A=1234.....  
B=A^&HF.....
```

の例は、A という変数に“ 1234 ”という値を代入しています。

の例は、A の値と 0F との間に 16 進で論理積をとっています。

2) 変数

変数の表現は、アルファベットの A ~ Z 迄の 26 文字とそれに数字を加えたものです。

```
A A0 A1 A2 ... A9  
B B0 B1 B2 ... B9  
      |  
Z Z0 Z1 Z2 ... Z9
```

以上 286 個の変数が使用できます。

3) 配列

P版では、A R(3 1), X(3 0 0), Y(3 0 0), Z(3 0 0), U(3 0 0), M(5 9 9 9) の 6 個の配列が使用可能です。Z版では、A R(3 1), X(2 5 6), Y(2 5 6), Z(2 5 6), U(2 5 6) の 5 種類です。

4) 関数

関数は、入力系のコマンドに利用されています。例えば、S W(), I N() は入力ポートの状態を数値として表現します。関数は、変数としてそのまま扱うことができます。

```
A=SW(12).....SW(12)のポートの値をAに代入します。(1ビット入力)  
B=IN(0).....入力ポート0~7までを1byteの平行データとして読み取る。
```

5) 演算

演算・代入は、式により実施されます。

```
代入: A=1234  
      B=IN(0)  
      C=&HFF00
```

```
演算: D0=A+B  
      D0=123+B
```

演算は、前記の様に 2 項演算に限定されています。演算の種類は、加減乗除・論理和・論理積の 3 つがあります。

加	:	+	$A = B + C$
減	:	-	$A = B - C$
乗	:	*	$A = B * C$
除	:	/	$A = B / C$
余	:	%	$A = B \% C$
論理積 (AND)	:	^	$A = B \wedge C$
論理和 (OR)	:		$A = B \vee C$
排他的論理和 (XOR)	:	x	$A = B \times C$

$A=SW(0)\wedge B, A=SW(0)|B$ ということも出来ます。

6) 制御文

GOSUB	サブルーチンコールでRETURNにより戻ります。
GOTO	制御ジャンプです。与えられた文番号やラベルへ制御を移します。
FOR ~ NEXT	決まった回数繰り返す制御文です。
*ラベル	*マークに続く文字列でラベル文となり、文番号に依存しないプログラムを作成できます。
IF ~ THEN	条件が成立するTHENの後ろに指定された文番号もしくはラベルに制御を移します。コマンドを記述することはできません。
IF ~ GOSUB	条件が成立するとGOSUBの後ろで指定された文番号やラベルのサブルーチンを実行します。
ELSE__THEN	IF文と組み合わせて使用します。IF文の条件が成立しなかった時にTHENの後ろで指定された文番号やラベルに制御を移します。(P版のみ)
ELSE__GOSUB	前記と同様ですがサブルーチンコールとなります。(P版のみ)
END	プログラムを終了します。メインタスク以外でENDを実行すると、タスクは停止します。

7) 実行中のエラー

MPC - 816はプログラム作成時にメッセージにより各種エラーを表示しますが、プログラム実行中のエラーはMPC - 816の赤いLEDにて表示されます。バッテリーエラーやプログラム破壊などの致命的な場合は赤いLED点灯、実行中のエラーには赤いLEDが点滅します。いずれにしてもプログラムは、実行されませんのでパソコンに接続の上原因を確かめて下さい。また、エラー表示を出力ポートに引き当てることもできます。

参照コマンド ERR_ON

5.2 コマンド・リファレンス

【*ラベル】 機能：制御文

書 式 *ラベル

解 説 IF ~ THEN * LABEL , GOSUB * SHORI , GOTO * SAGYOなどの記述が可能です。また、LIST時にもLIST * COMMENTとすれば指定場所よりリストする事が出来ます。ラベル文は*を先頭とする文字列です。

文番号 * LABEL (10文字以内)

通常のリスト表示の場合にはラベルのみ表示しますが、FTMWより番号付でセーブされたファイルには文番号と併せて保存されます。そしてラベル表示は「」で囲まれます。

FTMW上の表示

```
100 *LOOP
110 A=A+1
120 IF A=100 THEN *LOOP
```

文番号付ファイルの場合

```
100 *LOOP
110 A=A+1
120 IF A=100 THEN [*LOOP]100
```

FTMW上でラベルを使用する時、あらかじめラベルが行き先に存在していなければなりません。例えば、*LOOPがまだ入力されていないのにGOTO *LOOPと記述することはできません。

【A__START】 機能：制御文

書 式 A_START 0
A_START n
n : 時間(sec)
10 n 137

解 説 A__START 0の場合 プログラミングケーブル接続時にはプログラムモードになる
A__START nの場合 プログラミングケーブルが接続されていてもパソコン側からの呼びかけがn秒以内になればプログラムが実行される

MPC - 816はプログラミングケーブルが接続されているとターミナルモードに、ケーブルを抜くか、接続せずにパワーオンすると自動的にプログラムが実行されます。ケーブルが接続されていてもA__STARTコマンドにより一定時間経過後にプログラムを実行することも可能です。

A__STARTで自動実行を行うときはプログラムにA__STARTコマンドを追加した後1度プログラムを実行して下さい。

```
10 A_START 20
20 *LOOP
30 FOR I=0 TO 255
40 OUT I,0
50 TIME 50
60 NEXT I
70 GOTO *LOOP
```

A__STARTを追加したらRUNコマンドかケーブルを抜いてプログラムを実行する。その後パワーオンリセットかリセットスイッチを押すと20秒後に自動実行。

【ACCEL】 機能：パルス

書式 ACCEL n[,m,i]
 n：最大スピード(pps)
 m：加速距離(pIs)
 i：オフセットスピード(pps)
 200 n 30000(MODE 5)
 2000 n 50000(MODE 6)
 m 5000 (省略するとm=n/10)
 0<1/2n

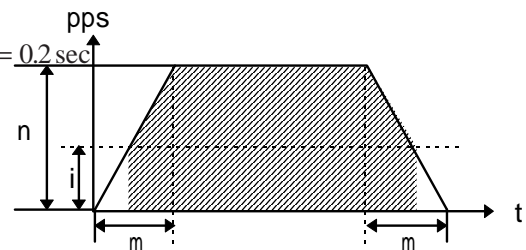
n及びmの下2桁は無効です。精度は、100毎です。また、FEEDのスピード値も正確なものではありません。

解説 n,mでFEEDの最大スピードと加速度を定めます。加速距離は停止から最高スピードに到達します。例えば、ACCEL 4000,400,0とした場合、加速距離は400パルスですから、1-2相駆動のステップモータにて約1回転で4000ppsに到達することになります。これを加速度に換算すると次の様になります。

$$\text{加速度} = \frac{n^2}{2 \times m} \text{ pps} / (\text{sec})^2 \quad \text{この場合では、} \frac{(4000)^2}{2 \times 400} = 20\text{k} / (\text{sec})^2 \quad \text{となります。}$$

また、最高速への到達時間は $t = \frac{2 \times m}{n}$ で、この例では、 $t = \frac{2 \times 400}{4000} = 0.2 \text{ sec}$ となります。

最後の引き数 i は、立ち上がりの最少スピードを定めます。ステップモータでは初速(自起動)を低く抑えすぎると振動等が発生するため、あるパルス数以下を出力しない様にします。これをスピードの変化を表したグラフにすると次の様になります。nとmで加速度を定め、iは最初の低速域をなくしています。ACCEL命令では加減速のパターンを定めると同時に、スピードも定めます。これは、次の式で表されます。



$$\text{FEEDは} \quad \text{FK} = n - \frac{n-1}{16} \times k \quad 0 \leq k \leq 15$$

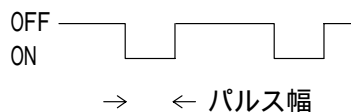
ACCELコマンドはRAM上に加減速テーブルを作成しますが、演算に若干時間がかかります。時間は指定されたパラメータにより変わり、緩やかな加減速ほど時間がかかります。ACCELコマンドはプログラムの先頭で行い途中でMOVEや、JUMPのスピードを変えるにはFEED、FEDZなどを使用した方がタイムロスが少なく済みます。

```
10 MODE 5
20 ACCEL 30000
30 *LOOP
40 FEED 0
50 MOVE 10000,10000
60 TIME 100
70 FEED 7
80 MOVE 0,0
90 GOTO *LOOP
```

ACCEL n(最大スピード)の設定値とパルスレートの関係

MPG - 303からパルス発生				
n	MODE5		MODE6	
	MOVE	JOG	MOVE	JOG
50000			56.3	14.6
45000			51.6	13.7
40000			44.2	12.0
35000			38.6	10.8
30000	29.0	13.7	32.5	9.3
25000	26.5	12.0	26.8	7.9
20000	21.1	8.9	21.3	6.4
15000	15.7	6.2	15.8	4.9
10000	10.4	3.9	10.3	3.3
8000	8.4	3.0	8.2	2.6
4000	4.1	1.4	4.1	1.3
2000	2.0	0.7	2.3	0.8
パルス幅	20 μ sec		4 μ sec	

MODE 6ではn < 2000の設定はできません。



極低速のパルス出力方法

MODE 5、MODE 6のACCELコマンドの最大スピードレートnの最小値は200までですが実際にはMODE 6では0.2kppsまでパルスレートは落ちません。MODE 6はサーボモーターを対象に設計されており低速には弱くなっています。またMODE 6はパルス幅も狭く、ステップモーターでは動作しないものがあります。次に極低速の出力方法の例をいくつか紹介します。

一般的FEEDコマンドによる方法です。

```

10  MODE 6
20  ACCEL 10000
30  FEED 0
40  FOR I=0 TO 5
50    MOVE 10000,0
60    MOVE 0,0
70  NEXT I
80  FEED 15          <--FEEDを最低速
90  GOTO 40

```

MODEを切り替えてACCELを設定しなおしています。

MODE 5, ACCEL 200の設定にはほとんど時間がかかりませんが、再びMODE 6, ACCEL 10000に設定するときは時間がかかります。

```

10  MODE 6
20  ACCEL 10000      <-時間がかかる。
30  FEED 0
40  FOR I=0 TO 5
50    MOVE 10000,0
60    MOVE 0,0
70  NEXT I
80  MODE 5          <-MODEを替える
90  ACCEL 200      <-この程度ならアッというまに終わる。
100 GOTO 40

```

RMOVコマンドを使って1パルスずつ出力する。

RMOVの後にタイマーを用いればどんなに遅いパルスも出力できます。

```

10  MODE 6
20  ACCEL 10000
30  FEED 0
40  FOR I=0 TO 5
50    MOVE 10000,0
60    MOVE 0,0
70  NEXT I
80  RMOV 1,0       <-RMOV で1発ずつ出力する
90  GOTO 80

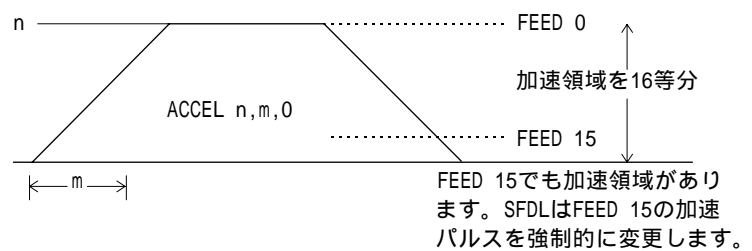
```

MODE、ACCEL設定を変えずにスピードを落とす方法としてSF DLコマンドを用いることもできます。これはFEED 15の値をさらに下げますが、MODE 6ではあまり効果ありません。

```

10  MODE 5
20  ACCEL 10000
30  FEED 0
40  FOR I=0 TO 5
50    MOVE 10000,0
60    MOVE 0,0
70  NEXT I
80  SF DL 1
90  FEED 15
100 GOTO 40

```



PULSEコマンドで低速パルスを出力することもできます。しかし、このコマンドは座標管理を行えません。

【A D】 機能：M I F - 8 1 6 A D

書式 AD(n)

n: 入力 i n 番号
0 n 3

解説 M I F - 8 1 6 A DのA / Dコンバータより計測データを取り出します。0 ~ 2がJ 4の2 8 , 2 9 , 3 0に対応します。返される数値は、0 ~ 4 0 9 5で1 m V単位です。

```

100 V=AD(0)
110 PRINT V

```

【A R】 機能：演算

書式 AR(n)

n: AR(n)の呼び番
0 n 31

解説 他の変数と同様、演算、代入等が可能。A R<ent>とすると配列の内容を表示します。

```

AR(0)=IN(0)
AR(1)=5
PRINT AR(0)

```

AR(n)はSF TR, SF TLコマンドでデータのシフトができます。

配列変数AR(n)とSF TRの使用例

```

10  FOR I=0 TO 7
20    AR(1)=I  <-配列変数 AR(0) ~ AR(7)にデータを代入。
30  NEXT I
40  '
50  *LOOP
60  PRINT AR(0),AR(1),AR(2)  <-AR(0) ~ AR(7)の内容を表示。
70  PRINT AR(3),AR(4),AR(5)
80  PRINT AR(6),AR(7)
90  PRINT
100 SF TR      <-右方向へ1回シフト。
110  AR(0)=AR(8)  <-はみ出た AR(8)(元 AR(7))を AR(0)に代入。
120  TIME 50
130  GOTO *LOOP
140  '
>RUN

```

```
0 1 2 ]
3 4 5 ]
6 7 ]
```

```
7 0 1 ]
2 3 4 ]
5 6 ]
```

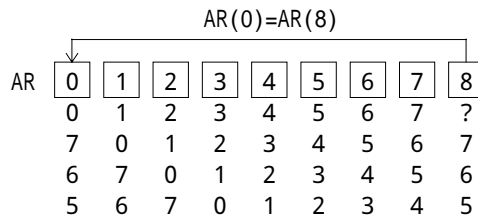
```
6 7 0 ]
1 2 3 ]
4 5 ]
```

```
5 6 7 ]
0 1 2 ]
3 4 ]
```

【シフトのイメージ】

配列変数は数値データを入れる箱です。SFTR、SFTLコマンドは箱の中身を1つずつ移しかえる作業を行います。

【SFTR】



【AR(n)にはWAITコマンドも使用できます。】

```
10     AR(0)=0
20     FORK 1,*JOB1
30     *LOOP
40     GOSUB *WAITSW
50     AR(0)=AR(0)+1
60     GOTO *LOOP
70     *JOB1
80     WAIT AR(0)=1
90     ON 0
100    WAIT AR(0)=2
110    ON 1
120    WAIT AR(0)=3
130    OUT 0,0
140    AR(0)=0
150    GOTO *JOB1
160    *WAITSW
170    WAIT SW(0)=0
180    WAIT SW(0)=1
190    RETURN
```

【Q】配列変数がAR(0)~AR(31)では足りない。

【A】このほか配列変数にはM(n)があります。さらにポイントデータX(n)、Y(n)、U(n)、Z(n)も配列変数として使用することができます。これらにはARにはあるローテート命令はありません。ポイントデータはNEWPコマンドで0に初期化されます。また、ポイントデータはFTMWでパソコンに読み書きできます。

```
10     FOR I=100 TO 105
20     X(I)=I+10
30     Y(I)=I+100
40     PRINT I,X(I),Y(I)
50     NEXT I
60     FOR I=0 TO 5
70     J=I+100
80     U(J)=X(J)+Y(J)
90     PRINT U(J)
100    NEXT I
```


【BKCNT】 機能：デバッグ

書式 BKCNT n
n：ブレークまでの回数
1 n 128

解説 BRKが、何回目で有効になるかを規定次の様なプログラムでI/Oの状態がどのような状態にあるかを確認したい場合

```
100 FOR I=1 TO 30
110 ON 1
120 TIME 100
130 OFF 1
140 IF SW(1)=1 THEN 160
150 ON 2
160 NEXT 1
```

```
100 FOR I=1 TO 30
110 ON 1
120 TIME 100
130 OFF 1
135 BRK
140 IF SW(1)=1 THEN 160
150 ON 2
160 NEXT 1
BKCNT 13
RUN
```

<-135 BRKを挿入してデバッグする。

前記の様にBKCNTを設定してBRKを挿入すると、指定の回数で停止します。BKCNTは通常1に設定されています。

【BL_AND】 機能：I/O

書式 BL_AND A1,A2,n
A1,A2：入力ポート、メモリーI/O、変数、定数
n：出力ポート

“_”はアンダーバーです。

解説 A1とA2をANDして結果が1であればnをONします。結果が0ならばnをOFFします。次のプログラムはSW(0)がオンになった時だけ出力0をオン、SW(1)がオフになった時だけ出力1をオンするというプログラムです。BL_ANDを使うと条件分岐をせずに出力をON/OFFすることができます。

```
10 *LOOP
20 BL_AND SW(0),1 0
30 BL_AND !SW(1),1 1
40 GOTO *LOOP
```

前記のプログラムをIF文でかくとこんな感じ...

```
10 *LOOP
20 IF SW(0)=1 GOSUB *ON0
30 ELSE_GOSUB *OFF0
40 IF !SW(1)=1 GOSUB *ON1
50 ELSE_GOSUB *OFF1
60 GOTO *LOOP
70 *ON0
80 ON 0
90 RETURN
100 *OFF0
110 OFF 0
120 RETURN
130 *ON1
```

```

140 ON 1
150 RETURN
160 *OFF1
170 OFF 1
180 RETURN

```

【BL_OR】 機能：I/O

書式 BL_OR A1,A2,n
A1,A2：入力ポート、メモリーI/O、変数、定数
n：出力ポート

解説 A1とA2のどちらかが1であればnをONし、両方が0のときnをOFFします。

```

10 SETIO
20 BL_OR SW(16),SW(17),0 <-SW(16)かSW(17)のどちらかがONならば出力0をON、
30 PRINT O_SW(0) 両方がOFFならば出力0はOFF
40 TIME 5
50 GOTO 20
>RUN
0
0
1
1

```

IF文で書くとこんな感じ...

```

10 *LOOP
20 A=IN(2)^&H03
30 IF A<>0 THEN *ON
40 IF A=0 THEN *OFF
50 GOTO *LOOP
60 *ON
70 ON 0
80 GOTO *LOOP
90 *OFF
100 OFF 0
110 GOTO *LOOP

```

【BRK】 機能：デバッグ

書式 BRK

解説 プログラム中の停止箇所に挿入。BRKによって停止したプログラムはCNTで再開します。

【BSY】 機能：パルス

書式 BSY(n)
0 n 3

解説 MPG-303の動作状態を数値で返します。引数のnの値は次の意味を持ちます。
n=0：現在アクセス中のMPGを指定する。タスク0ではPGコマンドによって選択されたMPG。タスク1～3ではMPG#1、タスク4～7はMPG#2、タスク8～11はMPG#3となります。
n=1,2,3:それぞれの番号に対応したMPGを指定します。また、関数が返す値は次の意味を持ちます。
0：MPG動作中。ただしJOGコマンドをのぞく。
1：MPG正常終了、コマンド待ち状態。
15：STOP 1コマンドにより減速停止。
240：STOP 2コマンドにより急停止。
3：OVRUN設定によって停止。

この様に、BSY(n)関数によりMPGの状態を監視することができます。しかし、JOGコマンドだけは動作中でも1をかえしてしまいます。

【CMND】 機能：MPG - 301

書式 CMND code
code : MPGアドレスとX3202命令コード
&Haaxx
aa : MPGアドレス(省略時#1)
xx : 命令コード

解説 MPG - 301に搭載されているパルス発生IC「X3202」のコマンドを実行します。
MPG - 301のコマンド/関数 : CMND, REG(), REG3(), ST_REG
参照 : 「MPG - 301 詳細マニュアル」

CMND &H08 "定速連続駆動=>ｽﾍﾞｰﾄﾞ は起動周波数(REG 3)

【CNFG#】 機能：RS - 232C

書式 CNFG# A1,A2,A3,
A1 : モード設定 0 A1 7
A2 : パリティ選択 0 A2 3
A3 : ボーレート選択 1 A3 6

解説 外部RS - 232Cポートの設定を行います。
MPC - 816のJ1には、SG(7)・TX1(8)・RX1(9)の端子があり、コマンド制御可能なシリアルポートがあります。モデム制御端子はありませんので他の機器との接続は、3線結合として下さい。サポートコマンドには、次の様な物があります

INPUT#, PRINT#, PUT#, GET#(), TST#()
通信フォーマットの設定です

A1 : モード設定
0 : NP7B1S NP=ノパリティ
1 : NP7B2S P=パリティ有り
2 : P7B1S 7B=7ビット
3 : P7B2S 8B=8ビット
4 : NP8B1S 1S=1ストップビット
5 : NP8B2S 2S=2ストップビット
6 : P8B1S
7 : P8B2S

A2 : パリティ選択
0 : 偶パリティXON/XOFF 有
1 : 奇パリティXON/XOFF 有
2 : 偶パリティXON/XOFF 無
3 : 奇パリティXON/XOFF 無

A3 : ボーレート選択
0 : 禁止
1 : 19.2K
2 : 9600
3 : 4800
4 : 2400
5 : 1200
6 : 600
7 : 禁止

A 1 ~ A 3 を省略するとパラメータの一覧が表示されます。

```
CNFG# a1 a2 a3
if a1 = -1 then clr CHO : CNFG# -1
a1 for mode 0:NP7B1S 1:NP7B2S 2:P7B1S 3:P7B2S
           4:NP8B1S 5:NP8B2S 6:P8B1S 7:P8B2S
a2 prty+xon 0:EVEN 1:ODD <--XON/OFF 制御有り
a2 prty+xof 2:EVEN 3:ODD <--XON/OFF 制御無し
a3 for baud 0:INHB 1:19.2k 2:9600 3:4800
           4:2400 5:1200 6: 600 7:INHB
```

【CNT】 機能：デバッグ

書式 CNT

解説 【BRK】もしくは、トレース中の<Q>キーによる停止での再スタートに使用。但し、【CNT】はプログラムを修正すると使用できなくなります。

【CONT】 機能：タスク操作

書式 CONT n[,m,l]
n,m,l : タスク番号
P版 1 n 11
Z版 1 n 7

解説 PAUSEの解除、タスクnの停止を解除します。FORKされた後PAUSEされていないタスクをCONTすることはできません。無視されます。

【CSW】 機能：I/O

書式 CSW(n)
n : 入力ポート番号
0 n 255 I/O
-128 n -1 メモリーI/O

解説 入力ポートの読み取り、スイッチの値が切り替わるまで関数値は返されません。

```
10 A=CSW(1)
20 IF A=0 THEN ~
30 IF A=1 THEN ~
```

10行では、1がONからOFF、或はOFFからONに切り替わるのを待ちます。Aに入る値は、切り替わった後のポート値です。ここでは20行、30行でポートがONになったかOFFになったかで異なる制御をします。

【!CSW】 機能：I/O

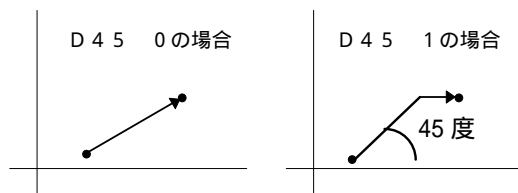
書式 !CSW(n)
n : 入力ポート番号
0 n 255 I/O
-128 n -1 メモリーI/O

解 説 入力ポートの読み取りですが入力ポートの値が変わるまで関数値は返しません。返す値はC SW(n)と
逆で、1が返されれば入力ポートがONからOFFへ、0が返されればOFFからONになったことを
意味します。

【D45】 機能：パルス

書 式 D45 n
n=0 : X・Y , Z・Uいずれも直線補完
n=1 : X・Y , Z・UいずれもD45移動
n=2 : X・YのみD45移動、Z・Uは直線補完
n=3 : Z・UのみD45移動、X・Yは直線補完

解 説 MPCのパルス発生では、直線補間パルス発生が基本となっています。この時ステップモータなどでX
Yテーブルを制御すると移動方向によって大きな振動音を発生させることがあります。これは、直線補
間での副軸側が指定したパルスレートより著しく遅いパルスレートとなることにより発生します。D4
5はこの様な場合に移動方向を45°と直線に分解して副軸の低パルスレートを避けます。この場合切
り換え点で移動は一但減速停止し、再スタートとなります。



【DA】 機能：MIF - 816AD

書 式 DA n
0 n 4095

解 説 MIF - 816ADのD/Aコンバータからの出力を変更します。J4 - 31に出力されます。出力も
1mV単位で、4.095Vまで出力出来ます。出力電圧は、AD(3)でモニタする事が出来ます。

DA 1000

1V出力します

【DEC】

FOR参照

【DEF__RST】 機能：I/O

書 式 DEF_RST A
A=&H00H~&HFFH ビットパターン

解 説 MPCリセット入力指定。MPCを外部入力でリセットするためのコマンドです。リセットとして使用
出来るのは入力0~7までの8点です。指定の方法はビットパターンAで、与えられた入力のどれかが
ONになるとソフトリセットが発生します。例えばDEF__RST &H11と入力すると入力0か4
のどちらかがONになるとソフトリセットします。指定されたポートはリセットされる毎に解除されま
す。このため、DEF__RSTコマンドはプログラム中に記述しておく必要があります。

10 DEF_RST &H01
20 *MAIN

【#DEFO】 機能：I/O

書式 #DEFO 文字列 番号

解説 #DEFSと同様の機能ですが、有効となるのはON/OFFの引数のみです。

```
10 #DEFO CYL1 2
20 ON CYL1
```

注) #DEFO、#DEFSともに、予め数字で入力されたプログラムに対しても有効です。プログラム終了後、このDEF文を追加することによって、全てのポート番号をシンボル化できます。定義されたポートは、シンボルでも数値でも入力できます。ここで定義されたシンボルが有効となるのは、ON、OFF及びO_SW(n)、ON_AND、OFF_AND等出力ポートのビット操作に関わるコマンドです。有効な文字数は8文字迄です。使用例は#DEFSと共通です。

【#DEFS】 機能：I/O

書式 #DEFS 文字列 番号

解説 SW(1)、SW(2)といったポート番号はプログラム保守時に意味が不鮮明となります。#DEFSは、この入力ポート番号を文字列で表現する為の定義文です。文字列は8文字以内です。

```
10 #DEFS SENS 1
20 IF SW(SENS1)=1 THEN ~
```

このポート番号の定義が有効なのは、HSW()、CSW()、!HSW()、!CSW()、WS()等の入力関数のみです。

```
10 #DEFO SOL1 0
20 #DEFS SENS1 0
30 ON SOL1
40 WAIT SW(SENS1)=1
```

定義ポート変更での注意は、#DEFO、#DEFSで一度定義したポート番号を変更しても出力コマンドや入力関数のパラメーターのポート番号は変わりません。

```
LIST 0
10 SETIO
20 #DEFS MSW1 -1
30 #DEFO SOL1 -1
40 ON SOL1
50 PRINT SW(MSW1)
60 OFF SOL1
70 PRINT SW(MSW1)
LIST 0
10 SETIO
20 #DEFS MSW1 -2 <-"-1"を"-2"に変更
30 #DEFO SOL1 -1
40 ON SOL1
50 PRINT SW(-1) <-"MSW1"ではなく"-1"になる
60 OFF SOL1
70 PRINT SW(-1)
>
LIST 0
10 SETIO
20 #DEFS MSW1 -2
30 #DEFO SOL1 -1
40 ON SOL1
50 PRINT SW(-2) <-"-1"を"-2"に変更
60 OFF SOL1
70 PRINT SW(-2)
>
LIST 0
10 SETIO
```

```

20 #DEFS MSW1 -2
30 #DEFO SOL1 -1
40 ON SOL1
50 PRINT SW(MSW1)      <- "MSW1"に戻る
60 OFF SOL1
70 PRINT SW(MSW1)
>

```

【DELETE】 機能：編集

書式 DELETE n,m
n,m : 文番号(存在する文番号)
0<n,m 32766

解説 削除する範囲を指定します。文番号 n より m までを削除。ある行より後を全て削除したい場合には、次のようにすると有効です。

```

32766 END
DELETE 100 32766

```

この様に、3 2 7 6 6 にダミーの行を入力しておいて削除します。また、次の様にもできます。

```

>TAIL
5000
>DELETE 100 5000

```

ここでは T A I L コマンドで最終行の内容を待ち、これにより全消去を実施しています。

【#DI】 機能：タスク操作

書式 #DI

解説 マルチタスクを発生させるタイマー割り込みを停止し、他のタスクを実行しません。マルチタスクへの復帰は、#EIを実行します。#DI実行後、TIME、SW()、FORK等タイマー、マルチタスク関係のコマンドを実行すると割り込み禁止は自動的に解除されます。INPUT文ではキャラクターが入っていないとやはりディスパッチしてしまいます。#DIを有効に使うにはHSWや演算のみを使い、タイマーやマルチタスク関係のコマンドを一切いれないことです。

【E】 機能：RS - 2 3 2 C

書式 E(n)
n=0

解説 P版ではRS - 2 3 2 C CH1のエラーを関数E(0)により知ることができます。これは外部装置との接続やON/OFFで発生するノイズキャラクターを検出し排除する場合に有効です。

```

戻り値0 エラー無し
1 パリティエラー
2 フレーミングエラー

```

エラー発生の場合は、CNFG#で初期化します。一週間に一度程度のRS - 2 3 2 Cエラーは自然に発生することも考えられますが毎日の様に発生すればそれは、ハード的に問題があります。RS - ISO等でノイズ源から分離して下さい。

【 # E I 】 機能：タスク操作

書 式 #EI

解 説 # D I によって停止されていたマルチタスクを再開します。

【 E L S E _ _ G O S U B 】 機能：制御文

書 式 ELSE_GOSUB N
ELSE_THEN N
Nはラベルまたは文番号

解 説 I F 文条件結果はその直後の行まで保存されています。(G O S U B 中の実行によっても破壊されません) また、 G O S U B によって R E T U R N した場合には、条件結果がスタックに積まれているため結果は破壊されず再利用することができます。

```
100 IF A=1 GOSUB *JOB1
110 ELSE_GOSUB *JOB2
(略)
200 *JOB1
210 RETURN
300 *JOB2
310 RETURN
```

上記の例では、 A = 1 の判断によって * J O B 1 か * J O B 2 のいずれかが実行され、 * J O B 1 を実行した場合は * J O B 2 が実行されることはありません。

【 E L S E _ _ T H E N 】 機能：制御文

ELSE_GOSUB参照

【 E N D 】 機能：制御文

書 式 END

解 説 プログラムの終了。各タスクの終了後には、必ず入力して下さい。 E N D 行が見つからないと下の行を次々と実行します。 E N D はタスクの停止も意味しています。メインタスク以外でプログラムの実行停止する場合は E N D 文が有効です。

```
10 FORK 1 *SUB
20 *LOOP
30 GOSUB *MAIN
40 GOTO *LOOP

100 *SUB
110 ON 10
120 TIME 100
130 OFF 10
140 END
```

<-ここで終了となる

【 E R A S E 】 機能：メンテナンス

書 式 ERASE

解 説 フラッシュROMのプログラムをクリアーします。プログラム編集後の R U N 、または F W R I T E によりフラッシュROMに書き込まれたプログラムをクリアーします。初期化コマンド「 M P C I N I T 」はフラッシュROMのプログラムはクリアーしません。 M P C I N I T をして S R A M のプログラムを

消去しても、次のパワーオン時にフラッシュROMに書き込まれたプログラムを展開します。全てのデータをクリアーする場合は「MPCINIT」ERASE」を連続して実行します。

```

10     FOR I=0 TO 7
20     ON I
30     TIME 10
40     OFF I
50     TIME 10
60     NEXT I
>RUN
Programming the FLASH ROM *+++++++   フラッシュROMに書き込みます。
>MPCINIT                               S R A Mの初期化
>ERASE                                  フラッシュROMプログラムクリアー
*                                       E R A S Eマーク
>

```

【ERR_ON】 機能：I/O

書 式 ERR_ON A
A：出力ポート番号

解 説 MPCのプログラム停止時のエラー表示。プログラムの実行中に致命的なエラーが発生した場合に指定した出力ポートをONにします。これは「PG IS OVRUN」,「RS-232C ERR」等のエラー発生時に外部にインタプリタが停止したことを知らせる機能です。ターミナル接続中にはコンソールへのメッセージ表示がありますが、自動実行中はインタプリタがただ停止するのみで外見としては何が発生しているのか不明です。このコマンドで出力ポートを指定しておく、MPC停止を外部で知る事ができます。ERR_ONが指定されていないとMPCの上の赤いLEDが一秒間隔で点滅します。また、この指定はリセット毎にクリアされてしまうので、プログラム中に記述しておく必要があります。また、エラー発生後エラー内容を確認するにはただちにターミナルを接続してMONコマンドを実行します。電源を入れ直して再実行したり、RUNをかけると、このデータは失われてしまいます。

ERR_ONコマンド使用上の注意

ERR_ONコマンドを用いた場合はタスク0をENDで停止しないで下さい。タスク0を停止すると監視プログラムが動作せず、エラー発生時に出力出来なくなります。

```

SETIO
ERR_ON 8                               <-エラーが発生するとON 8される様に設定
FORK 1,*TASK1
FORK 4,*TASK2
*LOOP0
TIME 5
GOTO *LOOP0                             <-ENDでタスク0を終了させない
*TASK1
MODE 5
ACCEL 30000,1000
OVRUN &H00FF
*LOOP1
RMOV 100000,100000
TIME 50
RMOV -100000,-100000
TIME 50
GOTO *LOOP1
*TASK2
MODE 6
ACCEL 50000,1000
OVRUN &H00FF
*LOOP2
RMOV 100000,100000
TIME 50
RMOV -100000,-100000
TIME 50
GOTO *LOOP2

```

【F E D D】 機能：パルス

書 式 FEED n
 0 n 15

解 説 ゲート・モーションには、Z軸の上昇と下降がありますが下降はワークの装着等に使用される為、上昇速度より遅く動作する必要があります。そのため、下降時のみ別に速度設定ができます。尚、F E D DはF E D Zより優先順位が低い為F E D Zの後に実行されます。F E D ZはF E D Dの値を自分の値と同一に設定します。これは、F E D Dの設定忘れを避ける為です。

【F E D H】 機能：パルス

書 式 FEDH n
 0 n 15

解 説 HOME命令で実行される相対移動の速度を設定します。この値は、リセットによって8に設定されます。

【F E D Z】 機能：パルス

書 式 FEEDZ n
 0 n 15 (PG 1~3)
 0 n 63 (PG -1)

解 説 U、Z軸の速度を選択します。FEED参照

【F E E D】 機能：パルス

書 式 FEED n
 0 n 15 (PG 1~3)
 0 n 63 (PG -1)
 0が最高速度

解 説 A C C E Lによって作られた加減速テーブルでどこまで加速するかを選択するコマンドです。通常A C C E Lによって最大使用レートを定め、プログラム実行中に速度を変更する場合にF E E D等を用います。F E E DはA C C E Lと異なり瞬時に作業を終了します。nで指定される速度についてはA C C E Lコマンドを参照して下さい。

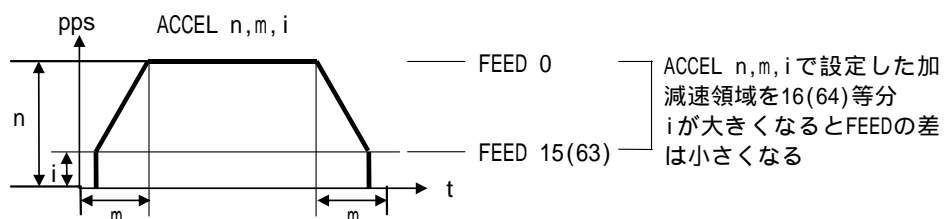
F E E D X、Y軸の速度を定めます。

F E D Z U、Z軸の速度を定めます。

F E D D JUMPでの下降移動の速度を定めます。F E D DはF E D Zの後に指定します。

F E D H 原点復帰コマンド実行中の原点よりの退避移動に対して有効です。

退避移動についてはHOMEを参照して下さい。



```

PG 1
MODE 6
ACCEL 20000,2000,5000 <-これは3秒ほどかかるので、ACCELコマンドはプログラムの
SETPOS 0,0             冒頭に設定して、実行中は変更しない。
*LOOP
FOR I=0 TO 15
FEED I                 <-スピードを変えるにはFEEDを使う。FEEDは時間がかからない。
MOVE 100000,100000
TIME 50
MOVE 0,0
TIME 50
NEXT I
GOTO *LOOP

```

HOME後のFEEDについての注意

HOMEコマンド実行後はFEEDを再設定してください。HOMEコマンドの退避移動のスピードを設定するコマンドとしてFEDHがあります。HOMEを実行するとFEEDの値はFEDHの値になります。次の様にACCEL設定後にFEED 0としてもHOMEコマンドを実行したあとのMOVEやRMOVのスピード(FEED)はFEDHの値になります。FEDHは初期状態で8ですから、HOMEの前にFEED 0とプログラムしてもHOMEを実行したあとはFEED 8になってしまいます。

```

>PG 1
>MODE 6
>ACCEL 10000

>FEED 0
>FEDH 8
>RMOV 20000,0          <-2.5sec    HOMEしないとFEED 0のスピードで動く
>HOME 0
>RMOV 20000,0          <-4.3sec    HOMEするとFEED 8で動く
>FEDH 0
>HOME 0
>RMOV 20000,0          <-2.5sec    FEDH 0のHOME後はFEED 0で動く
>FEED 15
>RMOV 20000,0          <-9.5sec
>FEDH 0
>HOME 0
>RMOV 20000,0          <-2.5sec    FEED 15でもFEDHが0ならスピードはFEED 0

>FEDH 8
>HOME 0
>FEED 0                <-HOME後にFEED 0設定
>RMOV 20000,0          <-2.5sec    FEED 0で動く

```

【FIND#】 機能：RS - 232C

書式 FIND# n
0 n &H7F(アスキーコード)

解説 nというASCIIコードを得るまで文字を読み捨てます。指定された文字は読み捨てません。バッファの先頭となります。AR()にデータを格納する場合、"A"という文字を先頭にするという条件が入ると次の様に記述できます。

```

10 CNFG# 2,0,2
20 FIND# 65
25 I=1
30 A=GET#(0)
40 IF A=13 THEN 1000
50 AR(1)=A
60 I=I+1
70 GOTO 30
1000 END

```

ここでは、CRコードが入力されると文字の取得を停止します。この様にTNYFSCでは、キャラクタを文字コード(数値)で扱う事によって文字列の処理を可能にしています。

【FOR .. NEXT / DEC】 機能：制御文

書式 加算 FOR a1=a2 TO a3 ~ NEXT a1
a1：変数
a2,a3：変数、定数 a2<a3
減算(P版のみサポート) FOR a1=a2 TO a3 ~ DEC a1
a1：変数
a2,a3：変数、定数 a2>a3

解説

```
*LOOP
FOR I=0 TO 7
  ON I
  TIME 10
NEXT I
'
FOR I=7 TO 0
  OFF I
  TIME 10
  DEC I
  GOTO *LOOP
```

FOR ~ NEXT / DECの途中からはGOTO、IF THENでは抜けられません

```
10     J=0
20     *LOOP
30     FOR I=0 TO 10
40         IF I=5 THEN *PASS
50     NEXT I
60     *PASS
70     J=J+1
80     PRINT J
90     GOTO *LOOP
>RUN
1
2
|
149
150
# 30
!! Out of Range
```

【FORK】 機能：タスク操作

書式 FORK n,m
n：タスク番号
1 n 11
m：文番号、または ラベル

解説 FORK文で起動されたプログラムは並列処理で実行されます。実行されているタスクをPAUSEやQUITしても出力やメモリーは初期化されません。また、タスクの状態は関数タスクで参照することができます。

```
10     FORK 1,*TASK           <-ラベル"TASK"からをタスク1で実行
20     INPUT A
30     PRINT TASK(1)         <-タスクの状態表示
40     INPUT A
50     PAUSE 1               <-タスクの一時停止
60     PRINT TASK(1)
70     INPUT A
80     QUIT 1                <-タスクの消滅
90     PRINT TASK(1)
100    END
110    *TASK
120    FOR I=0 TO 7
130        ON I
```

```

140    TIME 10
150    OFF I
160    TIME 10
170    NEXT I
180    GOTO *TASK
>RUN
?
0 <-タスク実行中
?
4 <-タスクが -ス
?
7 <-タスク消滅

```

すでに F O R K されているタスクをまた F O R K するとそのタスクは最初から実行されます。

```

10    *LOOP
20    FORK 1, *TASK1
30    TIME 50
40    GOTO *LOOP
50    *TASK1
60    FOR I=0 TO 100
70    PRINT I
80    TIME 10
90    NEXT I
100   GOTO *TASK1
RUN
0
1
2
3
4
0
1
2

```

<-FORKでタスク1は最初に戻る

【 F R E E 】 機能：編集

書 式 FREE

解 説 プログラムエリアの空き量をステップ数で表示します。NEW実行後、この量は最大書き込みステップ数となります。

```

>NEW
>FREE
2044

```

【 F W R I T E 】 機能：メンテナンス

書 式 FWRITE

解 説 フラッシュROMへプログラムを書き込みます。RUNと同様の書き込み動作ですが、プログラムを実行しません。

```

>FWRITE
*+++++++
>

```

フラッシュROMに書き込み中

【 G E T 】 機能：RS - 232C

書 式 GET(0)

解 説 RS - 232C CH0(プログラム用)に適應される他はGET #(0)と全く同等機能です。

【GET#】 機能：RS - 232C

書式 GET#(0)

解説 RS - 232C CH1のバッファより1byte取り出して関数値をそのキャラクタのASCII値として与えます。

例)CH1に外部ターミナルを接続し、次のプログラムを実行しターミナルでAのコードを出力(Aキーを押す)すると次の様に表示されます。

```
10 CNFG# 2,0,2
20 PRINT GET#(0)
30 GOTO 10
65
65
65
```

ここで、65は'A'=&H41のバイナリ値です。この様にGET#(0)是与えられた文字のコードを与えるため文字を捜す、或はある文字を変数に代入する事ができます。

```
100 FOR I=1 TO 10
110 AR(I)=GET#(0)
120 NEXT I
```

このプログラムデータは、AR()に次々とバッファの先頭からデータを取り込んでセットしていきます。点データを使用する場合はX(n)をディメンジョンとして扱う)

```
100 FOR I=1 TO 10
110 X(I)=GET#(0)
120 NEXT I
```

となります。前記がGET#(0)の使用方法ですが、うまく動作しない場合は最初のプログラムPRINT GET#(0)によって外部のデータがきちんと入力されているかどうかテストして下さい。尚、ターミナルが用意できない場合はJ1の8と9をショートしてPUT#によって入力にキャラクタを与えて下さい。

*CNFG#でXON/OFFを有効にすると&H13、&H11はとれません。

*RS - 232Cのバッファは64byteです。

GET#、GETN#の弱点

3byteの演算を確保するために乗除ルーチンの中で割り込みが禁止されています。このため演算をしながらの通信はきわめて不得意です。下のMASTERのプログラムではタイムアウトカウンタダウンやIF文の演算もありますが何よりまずいのは一文字受け取る毎にPRINT文で表示しています。PRINT文自体は問題がないのですが数値に展開するために相当の計算を内部で実施しています。このため割り込み禁止が多くなりすぎ、SLAVEのプログラムの様に連続でキャラクタが送出されている状況下では読みこぼしてしまいます。これを避けるにはMASTER1/MASTER2の様に通信中によけいなことをしない様にするか、通信速度をおとすしかありません。2400bpsでは問題にならない様です。スレーブ側のキャラクタ送出にタイマも入れても良いです。これと関連する問題として、GETN#()もキャラクタがおくられている時に実行すると読みこぼしをおこします。これはGETNが計算をしながら動作しているためです。これも通信中はRS()関数で様子をチェックしてからスタートすべきです。この問題をシステム側から改善するためには、演算の大幅な見直しが必要となること、また完全リエントラントな3byte乗除を8ビットマイコンの上で実行するとかなり低速になるという恐れもあります。たいへん申し訳ありませんが、MPCのRS - 232Cの受信については配慮をして使用するという事でお願ひします。

```
( MASTER )
*AG
A=49
CNFG# 4,0,2
PUT# A
C=10
```

```

*LP
GOSUB *GT
C=C-1
IF C>0 THEN *LP
TIME 100
GOTO *AG
*GT
T1=100
*GT1
IF RS(1)>0 THEN *GT9
T1=T1-1
TIME 1
IF T1>0 THEN *GT1
GOTO *RSE
*GT9
A8=GET#(0)
PRINT T1,RS(1),A8
IF A<>A8 THEN *RSE
RETURN
*RSE
PRINT STR(-1)
PRINT T1,RS(1),A8
END

( SLAVE )
*AG
CNFG# A=GET#(0)4,0,2
TIME 15
C=10
* PUT# A LP
C=C-1
IF C>0 THEN *LP
TIME 10
GOTO *AG

( MASTER1 )
*AG
A=49
CNFG# 4,0,2
PUT# A
C=10
WAIT RS(1)=10
*LP
GOSUB *GT
C=C-1
IF C>0 THEN *LP
TIME 100
GOTO *AG
*GT
T1=100
*GT1
IF RS(1)>0 THEN *GT9
T1=T1-1
TIME 1
IF T1>0 THEN *GT1
GOTO *RSE
*GT9
A8=GET#(0)
PRINT T1,RS(1),A8
IF A<>A8 THEN *RSE
RETURN
*RSE
PRINT STR(-1)
PRINT T1,RS(1),A8
END

( MASTER2 )
*AG
A=49
CNFG# 4,0,2
PUT# A
C=10
*LP
GOSUB *GT
C=C-1
IF C>0 THEN *LP
PRINT A
TIME 10
GOTO *AG
*GT

```

```

T1=100
*GT1
IF RS(1)>0 THEN *GT9
T1=T1-1
TIME 1
IF T1>0 THEN *GT1
GOTO *RSE
*GT9
A8=GETN#(0)

IF A<>A8 THEN *RSE
RETURN
*RSE
PRINT STR(-1)
PRINT T1,RS(1),A8
END

```

【GETN#】 機能：RS - 232C

書式 GETN# A

A：変数名

解説 RS - 232C CH1のバッファより数字列を取り出し、その値を変数に返します。最初の文字が数字、数字記号でないと0を返します。その場合、バッファの内容は減りません。数字以外の文字に出会うと処理を終了します。その場合のデリミタはバッファにのこります。GETN#は実行中に割り込み禁止の時間があります。このため連続的に数値を読み取る場合には次の配慮が必要となります。

```

10 GETN# A0
20 SKPSP#
30 GETN# B0
40 SKIP# &HOD

```

GETN#の割り込み禁止により
シリアルデータがバッファに入らないことがある

このプログラムでは、20及び30が正常に動作しない事があります。対策としてはTIMEもしくはRS(1)など次の様に組み合わせます。

```

10 WAIT RS(1)>2 <-キャラクタが入ってくるのを検出
20 TIME 10 <-通信が一段落するのを待つ
30 GETN# A0
40 SKPSP#
50 GETN# B0
60 SKIP# &HOD

```

バッファの中のデータを読むので
割り込み禁止の影響なし

【GOSUB】 機能：制御文

書式 GOSUB n

n：文番号(存在する文番号)またはラベル

解説 指定の文番号(ラベル)の所へサブルーチン・ジャンプします。サブルーチン・コールの為、RETURN文により戻る事が出来ます。

```

10 GOSUB 100
20 ON 3
30 END
100 ON 4
110 RETURN

```

この例では、100、110のプログラムがサブルーチンとなっています。10でサブルーチン100に移り、110のリターン文で戻り20番を実行しています。この様に、サブルーチンはまとまった単位を再利用するのに適しています。GOSUBは必ずRETURNで返して下さい。ネストが深くなりすぎるとエラーになります。


```

10 GOSUB 10
>RUN
# 10
!! Stack Overflow

```

【GOTO】 機能：制御文

書式 GOTO n
n：文番号(存在する文番号)またはラベル

解説 指定の文番号(ラベル)のところへジャンプします。
注)尚、GOTO文は指定の文番号を検索する為、実行スピードは最初の1回が遅くなります。2回目以降は、検索の結果を覚えており他のコマンドと同じスピードで実行されます。(GOSUB, IF~THEN~, IF~GOSUB~も同様)

【HCSW】 機能：I/O

書式 HCSW(n)
n：ポート番号
0 n 255

解説 入力ポート読み込み。CSW(n)が5msecのタイマーをとって検出するのに対してHCSW(n)はタイマーがありません。従って、高速で入力の変化を検出します。チャタリング、ノイズの多い環境での使用には適しません。

【HOM】 機能：パルス

書式 HOM

解説 4軸ロボット、或はX, Y, Z等のZ軸を含んだロボットを製作された場合に、移動点から安全に原点に退避するコマンドです。実際には、

```

MOVZ 0
MOVE 0 0 0

```

を連続して実行し、Z軸上昇、X, Y, U移動というシーケンスとなっています。

【HOME】 機能：パルス

書式 X, Y軸原点復帰
HOME n,x,y
n:XS1~YS2までのセンサーパターン
x,y:退避移動量
Z, U軸原点復帰
HOMZ n,z,u
n:US1~ZS2までのセンサーパターン
z,u:退避移動量

解説 X, Y軸の原点復帰を実施します。x, yの値は原点復帰に先立つ退避移動量です。そのスピードはFEDHによって決められます。デフォルトは8です。X, Yを略すと以前の値が使用されます。nはビットごとに次の意味を持ちます。
例えば、XS1が1、YS1が1になった時原点復帰が成立するには次の様にします。

	BIT0	BIT1	BIT2	BIT3
MPG-303(J2)	XS1	XS2	YS1	YS2

HOME 5 0 0

ノーマルオンでセンサ検出にオフになる様なセンサが4つの場合は次の様にします。

HOME 0 0 0

原点復帰の方向と速度を決めるのはSHOMコマンドです。 参照コマンド：SHOM

```
10 MODE 5          <-MODE設定が一番始め
20 OVRUN 0         <-オーバーラン設定 この場合はオーバーラン停止無し
30 ACCEL 30000     <-加減速テーブル作成
40 SHOM 1,0,100    <-原点復帰設定 X軸CW方向,Y軸は無し,速度353pps
50 FEDH 15         <-退避移動スピード設定 0(高速)~15(低速)
60 HOME 1, -1000,0 <-原点復帰 XS1がONまで,退避量X=1000mm/sCCW方向
```

HOME コマンド実行時の座標について

HOME中にターミナルから<CTRL>+<A>で停止をかけると退避量が座標値として残ります。

```
>HOME &HF 500 500    <-退避量500mm/sでX,Y軸の原点復帰実行
                    <-<CTRL>+<A>で停止する
      TASK 0 # 30
T
PG 1#1(X,Y,Z,U) 500 500 500 500 [XYZ,U] 400 400 <-退避量が座標値になる
```

HOMEが正常に完了したときやSTOPコマンドで停止したときは座標値は0になります。

```
>HOME 0 500 500     <-正常完了のHOME
T
PG 1#1(X,Y,Z,U) 0 0 500 500 [XYZ,U] 400 400

10 HOME &H000F,500,500
FORK 1,10           <-文番号10をヌリで実行
>STOP 1             <-ヌリから強制停止
>T
PG 1#1(X,Y,Z,U) 0 0 0 0 [XYZ,U] 400 400
```

HOMZ コマンドの実行後の座標値

TNYFSC(R) Rev-3.22f [VER-P max2044]
Copyright (C) by ACCEL CORP/BC-SOFT
[300point MPG-303(PG 1-3)MODE5]6]

U軸のみの原点復帰

```
>SETP 0, -500, -500  □ 現在点の座標値設定
>STPZU 0, -500, -500 □
PG 1#1(X,Y,Z,U) -500 -500 -500 -500 [XYZ,U] 100 100 <-確認
>SHMZ 1,0,1000      <-原点復帰パターン U: CW Z:無し スピート:1000
>HOMZ 1,0, -200     <-原点復帰 センサパターン:US1=ON 退避移動量:Z=0,U=-200
PG 1#1(X,Y,Z,U) -500 -500 0 0 [XYZ,U] 100 100 <-実行後の座標
```

Z軸のみの原点復帰

```
>SETP 0, -500, -500
>STPZU 0, -500, -500
PG 1#1(X,Y,Z,U) -500 -500 -500 -500 [XYZ,U] 100 100
>SHMZ 0,4,1000      <-原点復帰パターン U:無し Z: CW スピート:1000
>HOMZ 4, -200,0     <-原点復帰 センサパターン:ZS1=ON 退避移動量:Z=-200,U=0
PG 1#1(X,Y,Z,U) 0 0 0 0 [XYZ,U] 100 100 <-実行後の座標
```

U、Z同時に原点復帰

```
>SETP 0, -500, -500
>STPZU 0, -500, -500
PG 1#1(X,Y,Z,U) -500 -500 -500 -500 [XYZ,U] 100 100
>SHMZ 1,4,1000 <-原点復帰パターン U: CW Z: CW スピード: 1000
>HOMZ 5, -200, -200 <-原点復帰センサーパターン: US1, ZS1=ON 退避移動量: Z=-200, U=-200
PG 1#1(X,Y,Z,U) -500 -500 0 0 [XYZ,U] 100 100 <-実行後の座標
```

前記3パターンの原点復帰ではいずれも実行後のU、Zの座標値は0になります。Z軸のみの原点復帰ではX、Yの座標値も0になります。SHMZの原点復帰方向のU、Zのパラメーターの位置とHOMZの退避移動量のパラメーターの位置が反対です。注意して下さい。

省略可能なパラメーターについて

HOMEコマンドの第2(X軸退避量) 第3(Y軸退避量)パラメーターは省略可能ですが、その場合は以前に設定された数値が有効になります。プログラム中で省略してしまうと、そのうちにどのような設定がされているのか判らなくなったり、RAM化けをおこして設定値が変わったりしたときプログラムが正常に動作しなくなったりします。HOMEコマンドに限らず省略可能なパラメーターでもプログラムには全部記述しておきましょう。

```
HOME 1,500,500 <-ダイレクトコマンドで設定した値が有効になる
>100 HOME 1 <-プログラム中に書いておかないとそのうちわからなくなる。
                ポートを交換したりRAM化けを起こしたら期待通りに動作しなくなる。
```

【PG-1のとき】

PG-1でMIF-816からパルス発生する場合の原点センサーはMIF-816のI/Oポート(SW(16)~SW(19))へ接続します。このポートに原点センサー以外は接続しないで下さい。

ビット	BIT0	BIT1	BIT2	BIT3
MPG-303(J2)	XS1	XS2	YS1	YS2
MIF-816(J4)ピン	1(SW(16))	3(SW(17))	5(SW(18))	7(SW(19))

【HOMZ】 機能：パルス

書式 HOMZ n,z,u
n: US1~ZS2までのセンサーパターン
z,u: 退避移動量

解説 Z、U軸の原点復帰を実施します。z、uの値は原点復帰に先立つ退避移動量でそのスピードはFEDHによって決められます。デフォルトは8です。Z、Uを略すと以前の値が使用されます。nはビットごとに次の意味を持ちます。

	BIT0	BIT1	BIT2	BIT3
MPG-303(J2)	US1	US2	ZS1	ZS2

例えば、US1が1、ZS1が1になった時原点復帰が成立するには次の様にします。

```
HOMZ 5 0 0
```

ノーマルオンでセンサ検出にオフになる様なセンサが4つの場合は次の様にします。

```
HOME 0 0 0
```

原点復帰の方向と速度を決めるのはSHMZコマンドです。

参照コマンド: SHMZ, HOME

【HPT】 機能：I/O

書式 HPT(n)
0 n 8

解説 MPG-303の原点入力の状態を知らせます。例えば、PRINT HPT(0)として0が返されればXS~ZSまで全てOFFの状態です。nが0の場合はパラレルデータとして扱われますが、1~8の値をセットすればXS~ZSの値をそれぞれ単独で見ることができます。HPT(0)は、MPG-303が動作している時に使用することはできません。

センサー	XS1	XS2	YS1	YS2	US1	US2	ZS1	ZS2
nの値	1	2	3	4	5	6	7	8

```
PRINT HPT(1),HPT(2)
1 0
```

であればXS1がOFF、XS2がONの状態です。HPT(n)は実行されたタスクに対応するMPG-303の原点入力を与えますが、MPG-303動作中に使用することはできません。パルス発生するタスクと別にHPT(n)を使用する場合はパルス発生やACCELコマンドと同時に動作しない様にインタロックをとって下さい。

【HSW】 機能：I/O

書式 HSW(n)
n: ポートナンバー
0 n 255 I/O
-128 n -1 メモリーI/O
0~255

解説 SW(n)はポートの値を2度読みし信号が安定しているとそれを確定として、チャタリングなどによる誤判定を防止していますが、HSW(n)は実行された時のポート値をそのまま返します。ノイズの多い入力には使用しないで下さい。HSW(n)の多用は結果的にタスクの切り替えがひどく遅くなります。

```
10 A=HSW(0)
20 GOTO 10
```

という様なプログラムを2つFORKすれば1タスクの能力は本来の1/2、12本FORKすれば1/12になります。HSW(n)は特定のタスクでどうしても速く信号をひろいたい場合や、シングルタスクで使用する場合にはして下さい。

【!HSW】 機能：I/O

書式 !HSW(n)
0 n 255 I/O
-128 n -1 メモリーI/O

解説 HSW(n)の反転理論です。入力ポートオン時に0、オフ時に1を返します。

【HWS0】 機能：I/O

書式 HWS0(n)
HWS1(n)
n:ポート番号
0 n 255 I/O
-128 n -1 メモリーI/O

解説 タイム・アウト付き、ウェイト関数。インタロックとしては、次のコマンドと同様です。
HWS0(n) ~ WAIT HSW(n)=0
HWS1(n) ~ WAIT HSW(n)=1
HWS0(n), HWS1(n)は関数として用います。タイム・アウト機能が追加されており、タイム・アウトが条件成立で関数から抜けられます。WAITの条件が成立すれば0、タイム・アウトとなれば1の値を返します。タイム・アウトの時間(時間制限)は、TMOUで設定します。

```
10 IF HWS1(2)=1 GOSUB 1000
20 ON 3
30 IF HWS1(3)=1 GOSUB 1000
40 OFF 3
50 GOTO 10
1000 'ERROR'
```

前記の例は、条件に従って入力がON/OFFすれば10~15を順に実行しますが、何らかの事情でセンサーの入力がタイム・アウトとなると1000のエラー処理に移ります。

【HWS1】 機能：I/O

HWS0参照

【IF~THEN/GOSUB】 機能：制御文

書式 IF~条件式~THEN/GOSUB n
n:文番号 1 n 32766またはラベル

解説 IF文に続く式を評価し、条件が成立すればTHEN文に続く文番号へジャンプ。条件式は次の物があります。A、Bは変数もしくは関数、定数です。

IF A<>B THEN/GOSUB	AとBが等しくない
IF A><B THEN/GOSUB	AとBが等しくない
IF A=B THEN/GOSUB	AとBが等しい
IF A>B THEN/GOSUB	AよりBが小さい
IF A<B THEN/GOSUB	AよりBが大きい
IF A=>B THEN/GOSUB	AよりBが小さいか等しい
IF A>=B THEN/GOSUB	AよりBが小さいか等しい
IF A=<B THEN/GOSUB	AよりBが大きい等しい
IF A<=B THEN/GOSUB	AよりBが大きい等しい

【IN】 機能：I/O

書式 IN(n)
n:ポート番号
0 n 31 I/O
-16 n -1 メモリーI/O
8255アドレス

解説 入力ポートの平行入力。IN(n)には5msecのフィルターはありません。次にバンクナンバーと入力ポートの関係を一部表に示します。

	入力(SW(n))	IN(n)
MPC-816	0 -> 7	0
	8 -> 15	1
MIF-816	16 -> 23	2
	24 -> 31	3
MIO-816#1	32 -> 39	4
	40 -> 47	5
MIO-816#2	48 -> 55	6
	56 -> 63	7
MIO-816#3	64 -> 71	8
	72 -> 79	9
MIO-816#4	80 -> 87	10
	88 -> 95	11
MIO-816#5	96 -> 103	12
	104 -> 111	13

使用方法としてはD S Wなど複数の入力を一括して読み込む場合に有効です。入力0～7までの2桁のD S Wの入力は次の様になります。

```
D1=IN(0)^&H0F
D2=IN(0)/16
D2=D2*10
D1=D1+D2
```

8 2 5 5 のアドレス指定

IN(n)のnに8 2 5 5のアドレスを与えて入力するもできます。この場合も5msecのタイマーは入りません。出力の状態も読み取ることができます。ただし、データは反転しています。

```
A=IN(&H48)
>PRX A
&HF0
>A=Ax&HFF
>PRX A
&H0F
>
```

【INPUT】 機能：RS - 232C

書式 INPUT A1[,A2,A3]
A1～A3：変数

解説 RS - 232C CH0(プログラムポート)からデータを入力します。
下記の様に、プログラム実行中にプログラムに数を引き渡すのに使用します。変数は1～3個で複数指定の場合はスペースで区切り、続けて入力します。

```
10 INPUT A1
20 PRINT A1
20 END
RUN
?123 (123はキーボードにより入力)
123
```

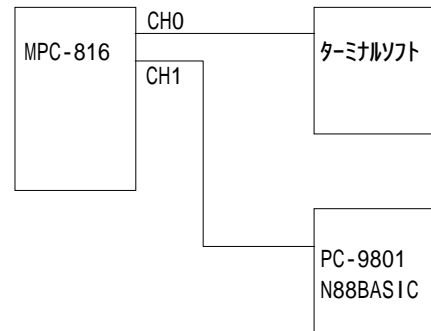
【INPUT#】 機能：RS - 232C

書式 INPUT# A1[,A2,A3]
A1,A2,A3 : 変数 / 定数

解説 RS - 232C CH1からデータを入力します。
受信処理は、数字、文字列とデリミタ・ターミネータに限られ次の書式となります。(デリミタは「スペース」(&H20)、ターミネータは「CR」(&H0D)です。)
入力書式 [数字文字列]スペース[数字文字列]CR]
[数字文字列]は3つまで入力可能です。尚、「LF」(&H0A)や余分なスペース等は無視します。INPUT#はINPUTと異なり受け取った文字列のエコーバックはありません。INPUT#は使用前にCNFG#でRS - 232Cの初期化を行って下さい。入力バッファは64byteです。

N88BASICプログラム

```
10 OPEN "COM:N81XN" AS 1
20 *LOOP
30 FOR A=0 TO 10
40 A1=A*100
50 PRINT #1,A,A1
60 PRINT A,A1
70 INPUT #1,B,B1
80 IF A<>B OR A1<>B1 THEN GOTO *AHO
90 NEXT A
100 GOTO *LOOP
110 *AHO
120 PRINT "???"
130 END
```



MPCプログラム

```
10 CNFG# 4,0,2
20 *LOOP
30 INPUT# A,A1
40 PRINT A,A1
50 TIME 50
60 PRINT# A,A1
70 GOTO *LOOP
>RUN
0 0
1 100
2 200
3 300
4 400
5 500
```

INPUT#ではCR受取後文字列の処理を実施しますがこの時演算中で割込禁止があります。このためCRに続いて文字列がさらに転送される場合は次の様にプログラムして下さい。

```
WAIT RS(1)>20
INPUT# A1
INPUT# B2
```

WAIT RS(1)>20で1回の転送文字数を予め持っておきます。ここでは、20文字以上としていますが、これは受け取る文字数に応じて変更します。

【 I O _ C N T 】 機能：ソフトカウンタ

書 式 IO_CNT

解 説 MPC - 8 1 6 の入力を使った 3 byte 長のカウンタで、3 チャンネルあり、バックグラウンドで動作します。in 0 , 2 , 4 をトリガ入力、in 1 , 3 , 5 を方向指示として使用し、変数 C 0 , C 2 , C 4 をカウンタとして使用します。このコマンドはマルチタスクでも有効ですが、早いカウントには追従できません(5 0 0 pps 程度まで)。エンコーダ(2 相クロック) をカウントするときは方向判別ユニットで信号を変換します。2 度読み等のチャタリング防止機能はありませんので注意してください。

カウンタ 0	in0 = count in1 = up/down	on でカウンタ C0 が変化 off で正/on で負
カウンタ 1	in2 = count in3 = up/down	on でカウンタ C2 が変化 off で正/on で負
カウンタ 2	in4 = count in5 = up/down	on でカウンタ C4 が変化 off で正/on で負

ソフトカウンタ関係のコマンド：IO_CNT, OUT_CNT, OUT_CSET

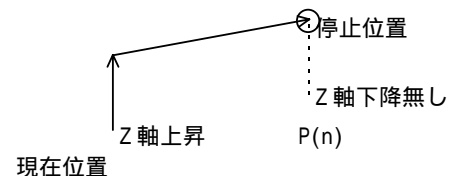
```
195 'COUNTER 1
200 SETVAR C0,C9,0
210 IO_CNT
220 PRINT C0,C2,C4
230 TIME 100
240 GOTO 220
```

【 J M P Z 】 機能：パルス

書 式 JMPZ P(n)

n : 点番号
1 n 300

解 説 現在位置より P(n) の上空への移動。ゲート・モーションですが最後の下降はしません。JUMP は、ゲート・モーションで門型の移動をしますが JMPZ では最後の Z 軸下降をしません。これは、ツーリング中の調整の便をはかる為設定されたコマンドです。



【 J O G 】 機能：パルス

書 式 JOG n,m

n : スピード
1 n 加速距離
m : 軸指定

1:XCW 2:XCCW 3:YCW 4:YCCW 5:UCW 6:UCCW 7:ZCW 8:ZCCW

(PG -1 では JOG コマンドは使えません)

解 説 JOG コマンドは、PG コマンドの中で唯一パルス発生中でもハングアップしないコマンドです。従って JOG コマンド発行後、そのタスクでただちに入力ポートのタイミング待ちによりパルスを停止することができます。JOG コマンドの有効範囲は X R A N G、Y R A N G、Z R A N G、U R A N G、コマンドで設定されます。JOG コマンドでは各 R A N G の上位 2 byte のみを比較して下位 1 byte は切り捨てられますから、殆どの場合、設定範囲いっぱいには動作しません。下位 1 byte を切り捨てると言うことはうちわで 2 5 5 パルスの範囲で R A N G が設定されます。次のプログラムの様に X R A N G を 1 0 0 0 (H E X で 3 E 8) に設定しても下位 1 byte が切り捨てられるので実際には 7 6 8 パルス(H E X で 3 0 0) しか動きません。


```

10 PG 1
20 MODE 5
30 ACCEL 1000
40 XRANG 1000, -1000 <- +1000, -1000で設定しても+768, -769まで。
50 *LOOP XRANG 1024, -1025と設定すると+1024, -1025まで動きます。
60 WAIT SW(0)=1
70 JOG 50, 1
80 WAIT SW(0)=0
90 STOP 2
100 TIME 10
110 PRINT X(0)
120 GOTO *LOOP
>RUN
768
768

```

動作範囲のMINが0の場合

次のプログラム の様に、XRANGの最小値(MIN)を0で設定しても実際は0座標まで動作しません。0まで動作させるにはMINを-1などとします。

```

10 PG 1
20 MODE 6
30 ACCEL 30000
40 SETPOS 1000, 0
50 XRANG 10000, 0 <- XRANGのMINが0だと255までしか動かない
60 WAIT SW(0)=1
70 JOG 200, 2
80 WAIT SW(0)=0
90 STOP 1
100 PRINT X(0)
>RUN
255 ←

```

```

50 XRANG 10000, -1 <- MINが-1~-256で-1座標まで動作します。
>RUN
-1

```

```

50 XRANG 10000, -256
>RUN
-1

```

```

50 XRANG 10000, -257
>RUN
-257

```

JOGの設定範囲について。

MODE 6でのJOG n, mのnの設定範囲は、最大5000まで、ACCEL n, m, iの加速距離m以下です。ACCEL設定時にm省略するとnの1/5までが設定範囲になります。

```

>ACCEL 10000, 5000, 1000 <-この場合JOGの加速距離は5000まで設定可能です
>JOG 5000, 1
>STOP 1
>JOG 5001, 1 <-5000を超えることはできません
!! Out of Range
>

```

```
>ACCEL 10000,2000,1000
>JOG 2000,1
>STOP 1
>JOG 2001,1
!! Out of Range
>
```

<-この場合JOGの加速距離は2000まで

```
ACCEL 10000
>JOG 2000,1
>STOP 1
>JOG 2001,1
!! Out of Range
```

<-mを省略するとnの1/5まで設定可能

```
>ACCEL 20000
>JOG 4000,1
STOP 1
>JOG 4001,1
!! Out of Range
```

```
>ACCEL 50000
>JOG 5000,1
STOP 1
>JOG 5001,1
!! Out of Range
>
```

<-nの1/5が5000以上でもJOGの加速距離は5000を超えることはできません

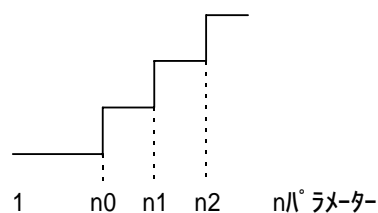
- * JOGコマンドに対してはBSY()関数は無効です。
- * JOGコマンドの停止はSTOP 1、2どちらも急停止となります。

パルスレート実測値

Rev 3.22g & MPG-303P '93/11/19
IWATSU SC-7201 UNIVERSAL COUNTER

次の表はJOGコマンドのパルスレートの実測値です。
JOGコマンドの演算は量子化されているため、段階的に変化します。特にMODE 6ではこのステップが荒くなっています。

SPEED



MODE 5

ACCEL の設定値	MOVE, RMOV コマンド MAX(KHz)	JOG コマンド	
		MAX(KHz)	MIN(KHz)
1000	1.0	0.3(200)	0.05(1)
2000	2.0	0.7(400)	0.05(1)
4000	4.1	1.4(800)	0.07(1)
8000	8.2	3.0(1600)	0.08(1)
10000	10.4	3.9(2000)	0.10(1)
15000	15.7	6.2(3000)	0.10(1)
20000	21.1	8.9(4000)	0.13(2以下)
25000	26.5	12.0(5000)	0.17(3以下)
30000	29.0	13.7(5000)	0.20(3以下)

JOGの()内はパルメータ設定値。最高ACCELの1/5以下で最大5000まで。

MODE 6

ACCEL の設定値	MOVE,RMOV コマンド MAX(KHz)	JOG コマンド	
		MAX(KHz)	MIN(KHz)
1000	2.3	0.8(200)	0.8(1)
2000	2.3	0.8(400)	0.8(1)
4000	4.1	1.3(800)	0.8(268 以下)
8000	8.2	2.6(1600)	0.8(133 以下)
10000	10.3	3.3(2000)	0.8(114 以下)
15000	15.8	4.9(3000)	0.8(76 以下)
20000	21.3	6.4(4000)	0.8(56 以下)
25000	26.8	7.9(5000)	0.8(48 以下)
30000	32.5	9.3(5000)	0.8(30 以下)
35000	38.6	10.8(5000)	0.8(23 以下)
40000	44.2	12.0(5000)	0.8(16 以下)
45000	51.6	13.7(5000)	0.8(16 以下)
50000	56.3	14.6(5000)	0.8(11 以下)

JOGの()内はパラメータ設定値。最高ACCELの1/5以下で最大5000まで。例えば ACCEL 40000 とすると JOG の最高速は JOG 5000,n で 12.0KHz、最低速は JOG 16 以下で 0.8KHz です。ACCEL が 1000 や 2000 では JOG のパラメータに関わらず 0.8KHz です。

【Q1】JOGコマンドは無限のパルス出力が可能か。

【A1】JOGコマンドはXRANG等のRANGコマンドで動作範囲の制限を受けるので無限のパルス発生はできません。

```

1      CHGREV 22
10     MODE 2
20     ACCEL 2000,1000,500
30     FEED 0
40     XRANG 32767,-32767      Z版では最高この範囲
50     SETPOS 0,0
60     *LOOP
70     IF SW(4)=1 GOSUB *JOG
80     WAIT SW(4)=0
90     GOTO *LOOP
100    *JOG
110    JOG 500
120    PRINT P(0)
130    RETURN
>
RUN
32767 0

```

XRANGで+ - 3 2 7 6 7を超えた設定をしようとするとエラーになります。

```

40     XRANG 40000,-32767
!!! Out of Range
???
```

うまく使えるかどうかはわかりませんがHOMEなら無限のパルス発生が可能です。しかしHOMEは減速停止しません。

```

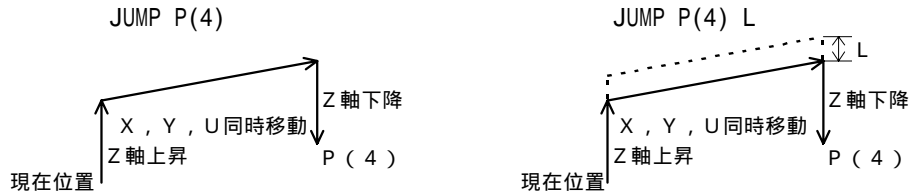
1      CHGREV 22
10     MODE 2
20     ACCEL 2000,1000,500
30     FEED 0
40     XRANG 32767,-32767
50     SHOM 1,0,200
60     HOME &H0003,0,0      入力16、17がONになるまでパルスを出します。

```

【 JUMP 】 機能：パルス

書 式 JUMP P(n)[,L]
n：点番号
L：Zの上限値、L=0であれば最上端まで上昇
1 n 300
0 L 8388607

解 説 現在位置よりP(n)へのゲート・モーション。ゲート・モーションとは、上昇、XY移動、下降という門型の移動です。



前記の図はゲート・モーションを表現しています。左の図は上限指定がない為、Z軸が0まで戻っています。右の図はLによって上限まで上昇せずに途中までとなっています。この為、この移動は移動時間が短縮されます。Z版モード1ではU軸をあわせた3軸同時移動となります。

【 KEY 】 機能：メンテナンス

書 式 KEY n
n：ロック番号

解 説 キーコマンドは、フラッシュROMにロックをかけて変更を阻止するコマンドです。

KEY 9812

とコマンドを実行すると9812をパスワードとしてロックがかかり、フラッシュROM上のデータは変更することができなくなります。パスワードは2byteの整数の範囲で設定してください。解除は同様に

KEY 9812

とします。番号を忘れてしまった場合は解除できなくなります。解除方法については各ユーザで一定の取り決めをして頂きシステム管理者として登録された方以外には公開できませんのでご了承ください。なお、キーコードに負の数を設定するとLIST表示もしくなくなります。

【 LET 】 機能：演算

書 式 LET ~算術式~ (LETは省略できます。)

解 説 T N Y F S Cでサポートされているのは、次の2項整数演算のみです。

A1=A2+A3	加
A1=A2-A3	減
A1=A2*A3	乗
A1=A2/A3	除
A1=A2%A3	余
A1=A2^A3	論理積(AND)
A1=A2 A3	論理和(OR)
A1=A2xA3	排他的論理和(XOR)(xは7E-ルックス)

注)オーバーフロー、アンダーフローの検出はサポートされていません。Xはスモールxです。P版では3byte整数、Z版では2byte整数での演算となります。

【LIST】 機能：編集

書式 LIST [n,m]
 n：開始文番号或は開始*ラベル
 m：表示行数
 0 n 32766
 1 m 127

解説 nを省略すると前回の続きを表示、mを省略するとLSCNTのnに従います。

LIST 30 <- プログラムを文番号30から表示
 LIST 30,5 <- プログラムを文番号30から5行表示
 LIST *MAIN <- ラベル*MAINから表示

【LOC】 機能：LCD

書式 LOC n,m[,s]
 n：行(縦) 1~2または4
 m：列(横) 1~16または20
 s：カーソル
 s=15：カーソル+プリンク
 s=14：カーソル
 s=13：プリンク
 s=12：カーソル無し

解説 n<>0の時はカーソル位置とカーソル形状の指定、n=0の時はmがLCD機種指定と画面消去になります。
 LCD機種(n=0)

m	LCD
0	BUS接続 4行20文字
1	MIF接続 4行20文字
2	BUS接続 2行16文字
3	MIF接続 2行16文字

LCD関係のコマンド：LOC,PRC,PRD,PRS

LOC 0,0 "LCD=BUS接続 4行20文字、画面クリア
 LOC 4,20,12 "カーソル位置 4行20文字、カーソル表示無し

注)LCDの接続、使用例については別途当社HP上のアプリケーションノートを御覧下さい。

【LP】 機能：ファイル

書式 LP n
 0 n 3

解説 SPコマンドでメモリエリアに保存した点データを取り出します。

【LSCNT】 機能：編集

書式 LSCNT [n]
n：表示行数
1 n 127

解説 標準リスト行数の設定、表示 n を省略すると現在の設定行数を表示

LSCNT 7 <--リスト行数を7行に設定
LSCNT <--リスト行数の表示

これは1画面で表示できる行数です。LISTは小さな画面でも使用し易い様にLSCNTで行数管理されています。次を続けて表示するにはLIST <ent>とします。最初から表示する場合は、LIST 0 <ent>とします。デフォルトは20です。

【M】 機能：演算

書式 M(n)

解説 M(0) ~ M(1199)
M(1200) ~ M(2399) 点データ#0と共通
M(2400) ~ M(3599) 点データ#1と共通
M(3600) ~ M(4799) 点データ#2と共通
M(4800) ~ M(5999) 点データ#3と共通

前記の様にM(0) ~ M(5999)までの配列が使用できますが、1200 ~ 5999まではSP nによってセーブされる点データエリアと共通になります。0 ~ 1199までは独立したエリアなので他の機能と干渉しません。

【MIO】 機能：I/O

書式 MIO n
n：816 or 240

解説 6枚目以降に使用するI/Oボードを選択します。初期値は下表「MIO - 248 モード」になっていますからMIO - 816を使用する場合はプログラム先頭に「MIO 816」と記述して下さい。

“MIO 248”モード			“MIO 816”モード		
ボード	OUT	IN	ボード	OUT	IN
MPC-816	0	0	MPC-816	0	0
MIF-816	8	16	MIF-816	8	16
MIO-816#1	16	32	MIO-816#1	16	32
MIO-816#2	24	48	MIO-816#2	24	48
MIO-816#3	32	64	MIO-816#3	32	64
MIO-816#4	40	80	MIO-816#4	40	80
MIO-816#5	48	96	MIO-816#5	48	96
MIO-248#1	112,56	(112)	MIO-816#6	56	112
MIO-248#2	136,80	(136)	MIO-816#7	64	128
MIO-248#3	160,104	(160)	MIO-816#8	72	144
MIO-248#4	184	(184)	MIO-816#9	80	160
MIO-248#5	208	(208)	MIO-816#10	88	176
MIO-248#6	238	(232)	MIO-816#11	96	192
			MIO-248#1	112	(208)
			MIO-248#2	136	(232)
			MIO-248#3	160	

()中の数値はMIO - 248の8点の入力ポート番号

【MODE】 機能：パルス

書 式 MODE n
 n : 5or6
(PG -1では無効です)

解 説 モードコマンドは、MPG - 3 0 3に収められたパルス発生モジュールを選択します。モードの選択は全ての命令に先だてて実施されなければなりません。加減速もスピード設定も与えられたモードに従って決定されます。モードを切り替えたらACCEL、OVRUN、HOME、HOMZ、SHOM等のパラメーターは全て見直さなければなりません。

```
100 MODE 5
110 ACCEL 30000
120 FEED 0
130 SHOM 1,4,200
140 HOME &HOA,100,100
```

5もしくは6で適切なものを選択します。MODE 5はステップモータ用、MODE 6はサーボモータ用と考えて下さい

MODE 5：最大パルスレート29.0kppsXY及びZUの2軸ずつの直線補間パルス発生機能を持ちます。パルス幅が20μsec確保されておりステップモータ用ドライバ向けの使用となっています。

MODE 6：最大パルスレート56.3kppsXY及びZUの2軸ずつの直線補間パルス発生機能を備えています。パルス幅が4.2μsecの為ステップモータドライバには不適です。

モード	パルス幅	デューティ比	最大パルスレート
5	20 μ sec	50%以上(*)	29.0kpps
6	4.2 μ sec	50%以下	56.3kpps

(*)但し、ACCEL 25000以下の場合

【MON】 機能：デバッグ

書 式 MON

解 説 BRK等のデバックはRUN命令によって実行されるメイン・プログラムのみ可能です。FORKによって実行されるプログラムは、デバックすることができません。その為、裏タスクで実行する前にRUNとBRKによってデバックし完全なものにしておく必要があります。又、実際の使用時にプログラムがハングアップしてしまい何処を実行しているのかがわからなくなることがありますが、こんな時はコントローラをリセット直後MONとします。

```
TASK 0 #10   TASK 1 #2000
```

前記の様なデータが表示され、停止していた場所が明らかになります。尚、プログラミング装置より実行中の場合は<CTRL> + <A>キーを入力すると実行中のプログラムが停止し、停止情報を表示します。また、FTMWでは<CTRL> + <M>によって対応する文番号のコマンドを表示します。

【MOVE】 機能：パルス

書 式 MOVE X,Y
 MOVE P(n)

解 説 絶対座標移動です。XY軸についてX、Yの位置までパルス出力します。ACCELによって最大スピードが決まりFEEDコマンドでスピードを切り換える事ができます。MOVE X1、Y1の場合のパルス発生量X2、Y2は

$$X2=X1-X(0)$$

$$Y2=Y1-Y(0)$$

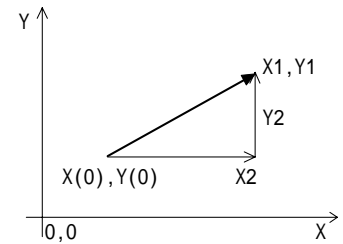
となります。P(n)を指定すると次と同じ意味となります。U及びZ座標は使用されません。

MOVE X(n),Y(n)

固定量パルスを出力する場合は、RMOVを使用して下さい。

途中で停止する場合はSTOPコマンドを使用します。

パルス発生にはMODE 5とMODE 6がありますが、MODE コマンドはいずれのモードでも同様に使用できます。異なった動作をします。選択についてはMODEコマンドを参照して下さい。



【MOVZ】 機能：パルス

MODE 5、MODE 6ともコマンド上の扱いは全く同等です。MOVZはMOVEコマンドがXY軸に対して有効なのに対してZU軸に有効になります。ACCELによって最大スピードが決まりFEDZコマンドでスピードを切り換える事ができます。

書式 MOVZ Z,U
MOVZ P(n)

解説 絶対座標移動ZU軸について、Z,Uの位置までパルス出力します。MOVZ Z1,Y1の場合のパルス発生量Z2,Y2は

$$Z2=Z1-Z(0)$$

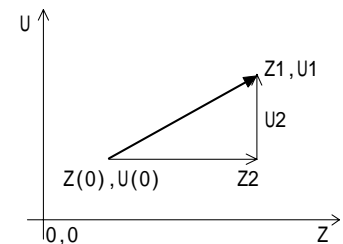
$$U2=U1-U(0)$$

となります。P(n)を指定すると次と同じ意味になります。X及びY座標は使用されません。

MOVZ Z(n),U(n)

固定量パルスを出力する場合は、RMOVZを使用して下さい。

途中で停止する場合はSTOPコマンドを使用します。



【MPCINIT】 機能：メンテナンス

書式 MPCINIT (ダイレクトコマンドでの使用のみ)

解説 P/Z版変更、システムアップデート後などにMPCINITを実行しRAMを初期化します。MPCINITが正しく実行されていないとプログラムがうまく入力出来なかったり、動作が異常になったりします。また、装置の調整にとりかかる前にMPCINITを実行すると工場出荷状態にもどります、組立配線時に静電気などでRAMの内容を壊してしまうことがあるため励行して下さい。プログラムには記述できません。

注意：MPCINITを実行すると点データやプログラムが壊されてしまいます。実行前にプログラムや点データはパソコンに保存して下さい。MPCINITはフラッシュROMに書き込まれたプログラムは消去しません。フラッシュROMの消去はERASEコマンドです。

【MTRX】 機能：パルス

書式 MTRX m,n
m,n:パレットの縦、横の数
1 m,n 32766

解説 mが点i~j、nが点i~kへの分割量 (PALET参照)

【MTRX1】 機能：パルス

書式 MTRX1 m,n
m,n:パレットの縦、横の数
1 m,n 32766

解説 mが点i~j、nが点i~kへの分割量 (PALET参照)

【NEW】 機能：編集

書式 NEW

解説 SRAM上のプログラムを消去します。新たにプログラムを作成する場合に使用します。ポイントデータの初期化はNEWPです。NEW後にFREEで表示されるステップ数は2044です。FROM上のプログラム消去はERASEです。

【NEWP】 機能：編集

書式 NEWP

解説 点データX(n), Y(n), Z(n), U(n)を全て0クリアします。

【NEXT】

FOR参照

【O_IN】 機能：I/O

書式 O_IN(n)
n:ポートバンク
0 n 31

解説 出力の状態をバンク単位で取得します。

```
>ON 0
>ON 1
>PR O_IN(0)
3
```

【O_SW】 機能：I/O

書式 O_SW(n)
n:出力ポート番号

解 説 出力ポートのON/OFFを読みだします。ON状態で1、OFF状態で0です。メモリI/OはSW(n)で、読み出すことがもともと出来ますのでMIOやその他の実際の出力ポートの状態を読み出すのに有効です。

```
ON 0
PRINT O_SW(0)
1
OFF 0
PRINT O_SW(0)
0
```

【OFF】 機能：I/O

ON参照

【OFF_AND】 機能：I/O

書 式 OFF_AND A1,A2,n
A1,A2：入力ポート、メモリI/O、変数、定数
n：出力ポート

解 説 A1、A2のANDの結果が1ならばnをOFFします。結果が0ならば出力状態は保持されます。ON_ANDコマンドの反対のコマンドです。

```
PR O_SW(0)
0 <-今出力0はONです
>OFF_AND 1 0 0 <-OFF_ANDの結果が0の時には出力は保持されます
>PR O_SW(0)
0 <-初のままです
>ON 0 <-出力0をONします
>PR O_SW(0)
1 <-確認
>OFF_AND 1 0 0 <-OFF_AND結果が0なので出力は保持されます
>PR O_SW(0)
1 <-なるほど初のまま
>OFF_AND 1 1 0 <-OFF_AND結果が1になると出力0はOFFされます
>PR O_SW(0)
0 <-初になった
OFF_AND SW(16),1,0 <-SW(16)がONならば出力0をOFFする
```

【OFF_OR】 機能：I/O

書 式 OFF_OR A1,A2,n
A1,A2：入力ポート、メモリI/O、変数、定数
n：出力ポート

解 説 A1、A2のどちらかが1であればnをOFFします。ON_ORコマンドの反対のコマンドです。

【ON】 機能：I/O

書 式 ON A1[,A2,A3]
OFF A1[,A2,A3]
A1,A2,A3：ポート番号
0 A1,A2,A3 255 I/O
-1 A1,A2,A3 -128 メモリI/O

解 説 ポートON/OFFはソレノイドやリレーのON/OFFに対応し、制御コマンドの基本です。尚、OFFはオープンコレクタ出力がOFF状態でレベルはHIGHとなり、ONはオープンコレクタ出力が

ON状態でレベルはLOWとなります。以上ON, OFFは初期出荷状態では負論理(ONにてシンクイン)と設定されていますがこれはSETIOコマンドによって変更する事ができます。(MIO-816 #1~#5迄のみ)

```

5 *LOOP
10 WAIT SW(1)=1
20 ON 2,3
30 WAIT SW(3)=2
40 ON 4
50 TIME 100
60 OFF 4
70 WAIT SW(1)=0
80 OFF 2,3
90 GOTO *LOOP

```

【ON_AND】 機能：I/O

書式 ON_AND A1,A2,n
A1,A2：入力ポート、メモリーI/O、変数、定数
n：出力ポート

解説 A1、A2のANDの結果が1ならばnをONします。結果が0ならば出力状態は保持されます。

```

>ON_AND 1 1 0          <-出力0をわします
>PR O_SW(0)
1
>ON_AND 1 1 0          <-ON_AND結果が1なので当然出力はわしたままです
>PR O_SW(0)
1
>ON_AND 1 0 0          <-ON_ANDは結果が0の時には出力は保持されます
>PR O_SW(0)
1
>BL_AND 1 0 0          <-OFFされません
>PR O_SW(0)
0

```

次のプログラムはSW(16)ともう一つのSWをみて両方がONならば出力をONします。

```

10 SETIO
20 WAIT IN(2)>1
30 ON_AND SW(16),SW(17),0
40 ON_AND SW(16),SW(18),1
50 ON_AND SW(16),SW(19),2
60 ON_AND SW(16),SW(20),3
70 PRINT O_SW(0),O_SW(1)
80 PRINT O_SW(2),O_SW(3)
90 WAIT IN(2)=1
100 GOTO 10
>RUN
1 0          <--SW(17)=1 SW(18)=0
0 0          <--SW(19)=0 SW(20)=0
0 1          <--SW(17)=0 SW(18)=1
0 0          <--SW(19)=0 SW(20)=0
0 0          <--SW(17)=0 SW(18)=0
1 0          <--SW(19)=1 SW(20)=0
0 0          <--SW(17)=0 SW(18)=0
0 1          <--SW(19)=0 SW(20)=1

```

A1、A2は定数や変数を、nにはメモリーI/Oを用いることができます。

```

10 SETIO
20 WAIT SW(16)=1
30 ON_AND SW(17),1,-1    <-SW(17)がONならばメモ-I/O-1をONする
40 ON_AND SW(18),1,-2
45 ON_AND SW(19),1,-3
50 ON_AND SW(-1),SW(-2),-4 <-メモ-I/O同士をANDしてメモ-I/OをON
55 ON_AND SW(-3),SW(-4),0 <-メモ-I/OをANDして出力0をON
60 PRINT O_SW(0)
70 WAIT SW(16)=0

```

```

80   GOTO 10
>RUN
1
1
0
0

```

この他に A 1 , A 2 には ! S W () , H S W () が使えます。

```

ON_AND !SW(16),1,0      <-SW(16)がOFFであれば出力0をON
ON_AND SW(16),!SW(17),8  <-とすればSW(16)がON、SW(17)がOFF状態の時
                          出力8がONされます。
IF SW(16)=1 AND SW(17)=0 THEN ON 8

```

【ON_OR】 機能：I/O

書式 ON_OR A1,A2,n
 A1,A2：入力ポート、メモリーI/O、変数、定数
 n：出力ポート

解説 A 1、A 2 のどちらかが1であればnをONします。OFF_ORコマンドの反対のコマンドです。

【OUT】 機能：I/O 種別：コマンド サポート：P・Z

書式 OUT n,m
 n：セットする値
 0 n 255 出力データ
 m：ポートバンク
 0 m 31 I/O
 -16 m -1 メモリーI/O

解説 データnをバンクmに出力します。ON/OFFコマンドは1ビットずつのオン・オフですがOUTは8ビットを1度のオン・オフします
 OUTコマンド使用例

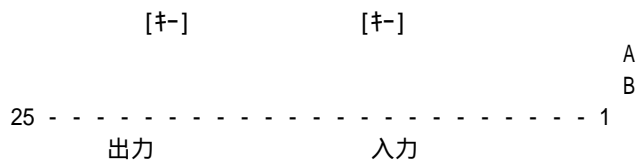
```

OUT 255,0      <-バンク0が全部ON
OUT 0,0       <-バンク0が全部OFF
OUT &HF,1     <-バンク1の下位4ビットがON
OUT &HF0,1    <-バンク1の上位4ビットがON

10 FOR I=0 TO 255
20 OUT I,0    <-バンク0にデータIを出力
30 TIME 10
40 NEXT I

```

MPC - 8 1 6、MIF - 8 1 6、MIO - 8 1 6 I/Oコネクタ配



MPC - 816シリーズ バンク/ポート表

コネクタ	ピン	MPC-816	MIF-816	MIO-816 #1	MIO-816 #2	MIO-816 #3	MIO-816 #4	MIO-816 #5	
J4	A1	0	16	32	48	64	80	96	入力
	A2	1	17	33	49	65	81	97	
	A3	2	18	34	50	66	82	98	
	A4	0 3	2 19	4 35	6 51	8 67	10 83	12 99	
	A5	4	20	36	52	68	84	100	
	A6	5	21	37	53	69	85	101	
	A7	6	22	38	54	70	86	102	
	A8	7	23	39	55	71	87	103	
	A9	8	24	40	56	72	88	104	
	A10	9	25	41	57	73	89	105	
	A11	10	26	42	58	74	90	106	
	A12	1 11	3 27	5 43	7 59	9 75	11 91	13 107	
	A13	12	28	44	60	76	92	108	
	A14	13	29	45	61	77	93	109	
	A15	14	30	46	62	78	94	110	
	A16	15	31	47	63	79	95	111	
B17	0	8	16	24	32	40	48	出力	
B18	1	9	17	25	33	41	49		
B19	2	10	18	26	34	42	50		
B20	0 3	1 11	2 19	3 27	4 35	5 43	6 51		
B21	4	12	20	28	36	44	52		
B22	5	13	21	29	37	45	53		
B23	6	14	22	30	38	46	54		
B24	7	15	23	31	39	47	55		

メモリー I/O バンク/ポート表

-1	-25	-49	-73	-97	-121
-2	-26	-50	-74	-98	-122
-3	-27	-51	-75	-99	-123
-1 -4	-4 -28	-7 -52	-10 -76	-13 -100	-16 -124
-5	-29	-53	-77	-101	-125
-6	-30	-54	-78	-102	-126
-7	-31	-55	-79	-103	-127
-8	-32	-56	-80	-104	-128
-9	-33	-57	-81	-105	
-10	-34	-58	-82	-106	
-11	-35	-59	-83	-107	
-2 -12	-5 -36	-8 -60	-11 -84	-14 -108	
-13	-37	-61	-85	-109	
-14	-38	-62	-86	-110	
-15	-39	-63	-87	-111	
-16	-40	-64	-88	-112	
-17	-41	-65	-89	-113	
-18	-42	-66	-90	-114	
-19	-43	-67	-91	-115	
-3 -20	-6 -44	-9 -68	-12 -92	-15 -116	
-21	-45	-69	-93	-117	
-22	-46	-70	-94	-118	
-23	-47	-71	-95	-119	
-24	-48	-72	-96	-120	

市販 D/A ボードの使用例

日本コムネット(株)製のD/Aコンバーター「GPY-15」の使用例です。GPY-15のアドレスは &H70 ~ &H73に変更してあります。

```

10 FOR A=0 TO 127
20 GOSUB [*DA]60
30 TIME 10
40 NEXT A
50 GOTO 10
60 *DA
70 A9=255-A
80 OUT A9,&H0070          <-アナログ出力
90 RETURN
100 *INIT
110 OUT &H0080,&H0073     <-GPY-15初期化
120 RETURN

```

市販 A / D ボードの使用例

日本コムネット(株)製 A / D コンバーター「GPY - 14」の使用例です。GPY - 14 のアドレスはデフォルトで &H40 ~ &H43 になっていますがこれは MIF - 816 と重なります。実際に使用するときは変更して下さい。

```

*LOOP
OUT 255,0
GOSUB *GPY
OUT 0,0
GOSUB *GPY
GOTO *LOOP
*GPY
OUT 0,&H43
TIME 1
A=IN(&H43)
PRINT A
RETURN

```

これらのボードを MPC ラックに挿入するにはアダプター「ADP - 325」が必要です。

市販ボードはアドレス &H50 以上で、I / O ボードなどと重複しない様にアドレスを変更して下さい。

【OUT_CNT】 機能：ソフトカウンタ

書式 OUT_CNT

解説 MPC - 816 の入力を使った 3 byte 長のカウンタで、3 チャンネルあります。変数 C1, C3, C5 に比較値を入れ、in0, 2, 4 をトリガ入力、in1, 3, 5 を方向指示として使用します。カウント値は変数 C0, C2, C4 に入り、比較値と一致すると out0 ~ 5 が変化します。このコマンドは条件成立して終了するまで、すべての割り込みが禁止され、シングルタスクになります。

カウンタ 0	in0 = count in1 = up/down	on でカウンタ C0 が変化 off で正/on で負
	out0,1	C0 <> C1 の間 on C0 = C1 で off
カウンタ 1	in2 = count in3 = up/down	on でカウンタ C2 が変化 off で正/on で負
	out2,3	C2 <> C3 の間 on C2 = C3 で off
カウンタ 2	in4 = count in5 = up/down	on でカウンタ C4 が変化 off で正/on で負
	out4,5	C4 <> C5 の間 on C4 = C5 で off

条件停止 in8 - in15 は out_cset で指定。

非常停止 in6 off でコマンド停止この時 out0-5 は off にする。

in7 on でコマンド停止この時 out0-5 は off にする。

ソフトカウンタ関係のコマンド：IO_CNT, OUT_CNT, OUT_CSET

```

90 'CONT CNTL
100 SETVAR C0,C9,0

```

110 C1=4
 120 C3=5
 130 C5=6
 140 ON 0,1,2
 150 OUT_CNT

注)方向判別ユニットと組み合わせて使用して下さい。応答パルスは1 kpps程度です。

【OUT_CSET】機能：ソフトカウンタ

書式 OUT_CSET wait mask rev
 wait : 出力ポートをオフしてからカウントのオーバーシュートを監視する時間
 mask : in8~15のビットパターン(例 in8, in9のみ監視は 0011b->3)
 rev : 1にしたビットを論理反転して検出

解説 OUT_CNTを動作条件を設定します
 ソフトカウンタ関係のコマンド : IO_CNT, OUT_CNT, OUT_CSET

OUT_CSET 100 5 4
 100 : 停止後1秒間カウンタを監視する
 5 : in8, in10のみ監視
 4 : n 10はB接点検出となる

【OVRUN】機能：パルス

書式 OVRUN n
 n=&H 入力24から31の入力パターン
 : 論理パターン
 : マスクパターン

解説



$J = (IN(3) \text{ XOR } \quad) \text{ AND}$

Jの値が0であれば停止しません。Jの値が0でなければ非常停止となります。論理パターンは各ビットを1(オン)で有効とするか0(オフ)で有効とするかを規定します。

OVRUN &H0FF <-ポート24から31のいずれかの入力ポートがオになれば非常停止
 OVRUN &HFFFF <-ポート24から31のいずれかの入力ポートがオになれば非常停止

パルス発生コマンド実行中にオーバーラン状態になった場合はそのコマンドでハングアップします。復帰はプログラムの再実行によります。

M P G - 3 0 3を2枚使用した時のオーバーランコマンドについて
 M P C - 8 1 6はオーバーラン状態になるとインタプリタが停止してしまいます。つまりM P G - 3 0 3を2枚使用している場合でもオーバーランになるとどちらも動作しなくなります。しかし、オーバーランの設定状態でそれぞれのM P Gの停止状態が若干異なります。次のプログラムはタスク1でM P Gの#1、タスク4でM P Gの#2の制御をしていますがO V R U Nの入力ポートの設定を&H000Fと&H00F0と違えてあります。このプログラムを実行して入力24~27のどれかをONするとタスク1のM P G #1は即停止しますが、タスク4のM P G #2は設定されたパルスを出し切るまで停止しません。入力28~31がオンの場合ではその逆になります。O V R U Nの設定を&H00FFにす

れば24～31のどれかがONした時点で#1、#2どちらも即停止となります。いずれにしろインタブリタは停止しますので再スタートするにはパワーオンリセットをしなければなりません。オーバーラン状態になった時でもインタブリタを止めずに制御を続行したい場合はOVRUNコマンドを使わずに監視タスクを設けSTOPコマンドでのパルス停止を行うこともできます。

```

    FORK 1,*TASK1
    FORK 4,*TASK2
*LOOP0
    TIME 5
    GOTO *LOOP0
*TASK1
    MODE 5                <-MPG#1のMODE設定
    ACCEL 30000,1000      <-MPG#1は24～27がONで即停止する
    OVRUN &H000F
*LOOP1
    RMOV 100000,100000
    TIME 50
    RMOV -100000,-100000
    TIME 50
    GOTO *LOOP1
*TASK2
    MODE 6                <-MPG#2のMODE設定
    ACCEL 50000,1000      <-MPG#2は28～31がONで即停止する
    OVRUN &H00F0
*LOOP2
    RMOV 100000,100000
    TIME 50
    RMOV -100000,-100000
    TIME 50
    GOTO *LOOP2

10    OVRUN &H01          <-SW(24)をオーバーラン入力設定
20    RMOV 100000,0
30    FOR I=0 TO 100
40      PRINT I
50      TIME 10
60    NEXT I
>RUN                                <-プログラムを実行してSW(24)がONするとプログラムは
                                        RMOVから抜けだし次のステップを実行。

    0
    1
    2
    3
    4
>                                <-CTRL+Aでプログラム停止
>PR SW(24)                        <-オーバーランセンサーが入った状態で再びRUNするとIra-表示
    1
>RUN
    # 20
    !! PG is OVRUN
>

```

OVRUN状態の表示

ターミナルパソコンを接続してRUNで実行した時

```
!! PG is OVRUN (*1)
```

ターミナルパソコンを接続しないで自動実行した時

```

ERR__ON設定無し          -->MPC - 8 1 6 赤LED点滅
ERR__ONで出力設定       -->指定出力ON (*1)

```

(*1)タスク0が動作していないと、表示・出力されません(タスクをENDで終了させない)

【 P 】 機能：パルス

書 式 P(n)
 n：点番号
 0 n 300

解 説 点データのベクトル形式です。内部ではX, Y, Z, Uの四次元ベクトルを意味していますが、コマンドによって使用される成分が異なります。MOVE、PRINTで直接扱うことができます。n = 0の時は、現在位置となります。

PRINT P(n)
 111 222

注)4次元の座標表示は、

PRINT P(n),Z(n),U(n)

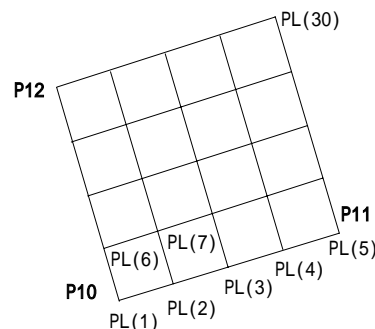
とする必要があります。P(n)は、SETP, STPZUで設定することが出来ます。また、Tコマンドによりパソコンによるインチング操作で表示することが出来ます。P版ではPLSコマンドにより一覧表示することも出来ます。

【 P A L E T 】 機能：パルス

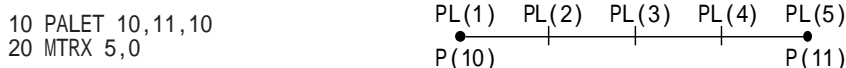
書 式 PALET i,j,k
 i,j,k：ティーチングされた点の番号
 1 i,j,k 300

解 説 この命令はセットで使用されます。PALETでパレットを決定する点を指定し、MTRXにてパレットの縦、横の数を定義しています。(次の図参照)

10 PALET 10,11,12
 20 MTRX 5,5



パレット宣言によってPL(1)~PL(30)の点を使用できるようになります。PALET, MTRXコマンドで作成された点は、PL(n), PLX(n), PLY(n)で使用することができます。(PALET 1に対応するのはPL 1(n), PL 1X(n), PL 1Y(n))PL(n)はP(n)と同じ4次元のベクトルデータとしてMOVE, JUMPコマンドで使用することができます。(U及びZについてはP(i)のUZ成分となる)各座標の値を必要とする場合は、PLX(n), PLY(n)を使用します。それぞれX成分、Y成分となります。1次元のパレットではiとkを同じ点番号とします。



【Q】PALET、MTRXコマンドの後にHOMEを実行するとパレットサイズの座標値はクリアされるか。

【A】PALET、MTRXの後でも先でもHOMEによる座標値クリアはありません。

10 PG 2
 20 MODE 5
 30 ACCEL 30000
 40 FEED 0
 50 SETP 1,0,0
 60 SETP 2,1000,1000
 70 SETP 3,1000,1000
 80 MOVE P(3)
 90 PRINT P(0)
 100 PALET 1,2,3

```

110 MTRX 11,1
120 GOSUB *HOME
130 'X=
140 PRINT STR(-1),X(0)
150 'Y=
160 PRINT STR(-1),Y(0)
170 '
180 FOR I=1 TO 11
190 MOVE PL(I)
200 PRINT PL(I)
210 NEXT I
220 END
230 *HOME
240 HOME 0
250 RETURN

```

```

>RUN
1000 1000 0 0
X= 0
Y= 0
0 0 0 0
100 100 0 0
200 200 0 0
300 300 0 0
400 400 0 0
500 500 0 0
600 600 0 0
700 700 0 0
800 800 0 0
900 900 0 0
1000 1000 0 0

```

```

FORK 4,10

```

```

>
1000 1000 0 0
X= 0
Y= 0
0 0 0 0
100 100 0 0
200 200 0 0
300 300 0 0
400 400 0 0
500 500 0 0
600 600 0 0
700 700 0 0
800 800 0 0
900 900 0 0
1000 1000 0 0

```

【Q】PALETコマンドは2つしか無く、PALET TO PALETの移動で足りなくなる

【A】PALET TO PALETなどの移動でPALETを複数使う場合でも移動前に再宣言すると、いくつものPALETに対応できます。時間的ロスも殆どありません。

```

10 PG 1
20 ACCEL 10000
30 FEED 0
40 SETPOS 0,0
50 '
60 SETP 1,0,0
70 SETP 2,1000,1000
80 SETP 3,1000,1000
90 SETP 11,2000,2000
100 SETP 12,4000,4000
110 SETP 13,4000,4000
120 '
130 *LOOP
140 FOR I=1 TO 11
150 GOSUB *PALET1
160 MOVE PL(I)
170 'PALET 1
180 PRINT STR(-1),X(0),Y(0)
190 '
200 GOSUB *PALET2
210 MOVE PL(I)
220 'PALET 2
230 PRINT STR(-1),X(0),Y(0)
240 NEXT I
250 GOTO *LOOP

```

```

260      '*****
270      *PALET1
280      PALET 1,2,3
290      MTRX 11,1
300      RETURN
310      '*****
320      *PALET2
330      PALET 11,12,13
340      MTRX 11,1
350      RETURN
>RUN
PALET 1 0 0
PALET 2 2000 2000
PALET 1 100 100
PALET 2 2200 2200
PALET 1 200 200
PALET 2 2400 2400

```

【PALET1】 機能：パルス

PALET参照

【PAUSE】 機能：タスク操作

書 式 PAUSE n[,m,1]
 n,m,1：タスク番号
 1 n,m,1 11

解 説 タスクnを一時停止する。CONTによってタスクn停止が解除されます。MPG-303を使用中のタスクにはPAUSEを実施しないでください。使用する場合はSTOPコマンドを実施してBSY(0)にてMPG-303の動作が終了している確認をする必要があります。STOP、BSY()また、PAUSEに先立ちSTOP 4にてMPG-303へのコマンド出力を停止しておく扱いになります。

【PG】 機能：パルス

書 式 PG n
 n:MPG-303ボードアドレス 1 n 3
 -1を指定するとMIF-816のJ5からパルス発生

解 説 MPCは、3枚のMPG-303を扱うことが出来ます。PGとタスクの関係は次の通りです。
 TASK 0 PG nコマンドにより操作するMPGを設定します。nの番号がショートピンの番号に対応します。
 TASK 1～3 MPG#1 DSWを1に設定したものです。(出荷時)
 TASK 4～7 MPG#2 DSWを2に設定したものです。
 TASK 8～11 MPG#3 DSWを3に設定したものです。
 Tコマンド下では、TABキーによりMPG-303を切り換える事ができます。PGコマンドはタスク0に対してのみ有効なコマンドです。

MODE 5	タスク0
PG 1	PGコマンドでMPGを切り替えてACCEL
ACCEL 5000	の設定やSTOPコマンドを実行しています。
FEED 0	
PG 2	
ACCEL 5000	
FORK 1,*PG1	
FORK 4,*PG2	
INPUT A	
PG 1	
STOP 1	
PG 2	
STOP 1	

```

WAIT BSY(1)<>0
WAIT BSY(2)<>0
QUIT 4,1
END
*PG1
MOVE 10000,10000
WAIT BSY(0)=1
TIME 10
MOVE 0,0
WAIT BSY(0)=1
TIME 10
GOTO *PG1
*PG2
MOVE 10000,10000
WAIT BSY(0)=1
TIME 10
MOVE 0,0
WAIT BSY(0)=1
TIME 10
GOTO *PG2

```

タスク 1

このタスクのパルス命令はM P G # 1 で実行される。

タスク 4

このタスクのパルス命令はM P G # 2 で実行される。

1 ~ 1 1 のタスクで P G コマンドを使うと、その効果は 0 に反映されます。1 ~ 1 1 のタスクでは P G を使わないで下さい。トラブルのもとです。

```

10 FORK 1,*TASK1
20 PG 2
30 TIME 50
40 ACCEL 30000
50 FEED 0
60 MOVE 1000,0
70 *TASK1
80 PG 1
90 END

```

<--TASK0でPG2を宣言しても80行のPG1が有効になってしまいます。

n = - 1 のとき

PG -1とするとパルス出力ポートはM I F - 8 1 6 の J 5 へ切り替わります。そのとき S W 1 6 ~ 2 3 は原点センサー、S W 2 4 ~ 3 1 は停止入力ポート(O V R U N)となります。S T O P コマンドはサポートしていません。M O D E コマンドは無効です。座標は 3 byte 長で、どのタスクからでもパルス発生が可能です。パルス発生中はマルチタスクが停止します。PG -1は< C T R L > + < A > でリセットされます。

```

PG -1 "MIFからパルスを出すモード"
ACCEL 3000 "ACCELは必須、MODEは不要"
FEED 0
FEDZ 0
OVRUN &HFF "SW(24) ~ (31)のどれかがONで停止"
'OVRUN &H01 "SW(24)がONで停止。使わないポートは一般入力OK"
'OVRUN &HFFFF "SW(24) ~ (31)のどれかがOFFで停止"
'OVRUN &HFF01 "SW(24)がOFFで停止。使わないポートは一般入力OK"
GOSUB *HOME
*MAIN
GOSUB *CW
GOTO *MAIN
*CW
'CW
PRINT STR(-1)
TIME 5
FOR IO=1 TO 5
RMOV 1000 1000
RMVZ 1000 1000
GOSUB *OVRUNCHK
TIME 10
NEXT IO
MOVE 0 0
MOVZ 0 0
GOSUB *OVRUNCHK
GOTO *__RTN
*OVRUNCHK "OVRUNチェック"
IF IN(3)<>0 THEN *OVRUNERR "OVRUNが&HFFの時"
'IF IN(3)<>255 THEN *OVRUNERR "OVRUNが&HFFFFの時"
GOTO *__RTN
*OVRUNERR
'OVRUNERR
'OVRUNERR

```

```

PRINT STR(-1)
END
*HOME
SHOM 1 4 400
HOME &HOF 1000 1000
'
'HOME COMP
PRINT STR(-1)
TIME 5
SHMZ 1 4 400
HOMZ &HOF 1000 1000
'
'HOMZ COMP
PRINT STR(-1)
TIME 5
GOTO *__RTN
'=====
*__RTN
RETURN

```

"原点復帰
"XCW YCW スピート
"停止ハターン SW(16)~(19)までON、退避移動CW 1000
"使わないハターンは一般入力OK

"UCW ZCW スピート
"停止ハターン SW(17)~(23)までON、退避移動CW 1000
"使わないハターンは一般入力OK

【PGS】 機能：パルス

書式 PGS(n)
0 n 3

解説 MPG - 3 0 3の状態をモニターする関数はBSY(n)ですが、エラー値は次の動作によって失われてしまいます。PGS(n)はMPG - 3 0 3に発生したエラーを次のエラーまで保持しています。

n = 0 タスクに対応するMPGの状態を得る
n = 1, 2, 3 MPG 1 ~ 3を指定する

【PL】 機能：パルス

書式 PL(n)
PL1(n)
n:パレット上の点番号
1 n 32767

解説 パレット上の点を指定。点P(n)と同様の扱いが可能です。番号は1から始まり、MOVE、PRINTで直接扱うことができます。Z, U成分はPALETコマンドでの始点のZ, U成分となります。

```

10 PARET 1 2 3
20 MTRX 5 5
30 FOR I=1 TO 25
40 MOVE PL(I)
50 GOSUB 1000
60 NEXT I
70 END

```

【PL1】 機能：パルス

PL参照

【PL1X】 機能：パルス

PLX参照

【PL1Y】 機能：パルス

PLX参照

【 P L S 】 機能：パルス

書 式 PLS n
n=点番号

解 説 点データP(n)の表示です。 P L Sで点1より P L S nで点nより表示します。途中で一時休止しますが、停止の場合は 'Q' を押し継続の場合はその他のキーを押します。

```
PLS
P(1): 0 0 0 0
P(2): 0 0 0 0
P(3): 0 0 0 0
P(4): 0 0 0 0
P(5): 0 0 0 0
P(6): 0 0 0 0
P(7): 0 0 0 0
P(8): 0 0 0 0
P(9): 0 0 0 0
P(10): 0 0 0 0
ok                                     <-Qで中断 それ以外のキーで継続表示
>
```

【 P L S _ M I F 】 機能：パルス

書 式 PLS_MIF

解 説 Z P L S , Y P L S , W P L S , V P L S , の出力先は通常“ out 0 ”ですが、このコマンドによって M I F パルスポートに変更することができます。この設定はパワーオンリセット、もしくは PLS_MIF -1で解除できます。

【 P L X 】 機能：パルス

書 式 PLX (n)
PLY (n)
PL1X (n)
PL1Y (n)
n:パレット上の点番号
1 n 32767

解 説 パレット点データの X , Y 成分を得ます。特定のパレット点のみ位置補正して使用する場合などに有効です。次の例は同様の意味を持ちます。

```
FOR I=1 TO 10
  X1=PLX(1)
  Y1=PLY(1)
  IF I=5 GOSUB *HOSE1
  MOVE X1 Y1
  GOSUB *PICK
  GOSUB *PLACE
NEXT I

*HOSE1
  X1=X1+50
  Y1=Y1-10
RETURN
```

【 P L Y 】 機能：パルス

PLX参照

【 P R 】 機能：デバッグ

書 式 PR A1[,A2,A3]
 A1,A2,A3：定数，変数

解 説 P R I N T の省略形です。

【 P R C 】 機能：L C D

書 式 PRC n[,m,l]
 n,m,l：キャラクタ(アスキーコード)

解 説 L C D にキャラクタを表示します。
 L C D 関係のコマンド：LOC,PRC,PRD,PRS

LOC 1 1
PRC &HB1 &HB2 &HB3 "1行1文字目から'アイ'を表示

【 P R D 】 機能：L C D

書 式 PRD n,m
 n：表示文字数
 m：変数

解 説 L C D に変数の数値を表示します。
 L C D 関係のコマンド：LOC,PRC,PRD,PRS

LOC 1 1
J=-9999
PRD 5 J "1行1文字目から'-9999'を表示
LOC 2 1
J=8888
PRD 5 J "2行1文字目から'8888'を表示(1文字目は空白になります)

【 P R I N T 】 機能：デバッグ

書 式 PRINT A1[,A2,A3] 省略形 PR
 A1,A2,A3：定数，変数

解 説 I / O ，変数値の表示。実行時のプログラムには用いられませんがデバッグ調整時に有効。

P R I N T P (n) 点データを表示
P R I N T S W (n) ポートの値を表示
P R I N T A 1 , B 1 変数 A 1 , B 1 の値を表示
P R I N T I N (n) パラレルデータの表示

【 P R I N T # 】 機能：R S - 2 3 2 C

書 式 PRINT# A1[,A2,A3]
 A1,A2,A3：変数 / 定数

解 説 R S - 2 3 2 C C H 1 からの数値出力です。
 デリミタは「スペース」(&H20)、ターミネータは「CR・LF」です。

PRINT# 123 <-"123"[CR][LF] と出力します。
ASCIIコード と言えば [&H31][&H32][&H33][&HOD][&HOA]

PRINT# A B <-"(A)"[スペース]"(B)"[CR][LF] と出力します。
ASCIIコード と言えば [(A)][&H20][(B)][&HOD][&HOA]
(A),(B)はそれぞれの変数の値を数字列化したものです。

PRINT# はターミネーターとして「CR・LF」を出力します。「CR・LF」を出力したくないときはPUTS#コマンドを使います。

PRINT#、PUTS#は使用前にCNFG#でRS-232Cの初期化を行って下さい。

【PRS】 機能：LCD

書式 PRS n
n：表示文字数

解説 LCDに直前のコメント文を表示します。
LCD関係のコマンド：LOC,PRC,PRD,PRS

```
LOC 1,2
'accel
PRS 5
```

"1行2文字目から'accel'(5文字)を表示

【PRX】 機能：デバッグ

書式 PRX A
A：変数、定数

解説 与えられた引き数をヘキサ表現で出力します。パラレル入力などをPRXで表示すればビットの状態がよくわかります。

```
PRX IN(0)
&HOF
PRX IN(5)
&HCO
```

【PULSE】 機能：パルス

書式 PULSE n[,D1,D2]
n：パルス発生数
D1：ON時間 単位100µsec
D2：OFF時間 単位100µsec
-32767 n 32767
1 D1,D2 32767
(PG -1では使えません)

解説 固定デューティのパルス出力コマンドです。D1、D2を省略すると2msecの周期でパルス発生します。D1、D2を10以下で使用すると誤差が大きいのので注意して下さい。出力する軸はAXISコマンドで設定します。PULSEコマンドでは座標管理を行いません。

```
AXIS 1 <-X軸指定
PULSE 1000 <-ON 1msec,OFF 1msecで1000CWAパルス
PULSE -1000,100 <-ON 10msec,OFF 10msecで1000CWAパルス
PULSE 2000,100,10 <-ON 10msec,OFF 1msecで2000CWAパルス
```


PULSEコマンドパルスレート

PULSE 1000	10 10	500.466Hz
	9 9	555.425
	8 8	623.943
	7 7	711.746
	6 6	828.307
	5 5	990.523
	4 4	1.232K
	3 3	1.628K
	2 2	2.401K
	1 1	4.573K

【PUT】 機能：RS - 232C

書式 PUT A1[,A2,A3]
 A1,A2,A3 : 変数 / 定数
 0 A1,A2,A3 &H7F

解説 A1, A2, A3の値をアスキーコードとしてCH0に出力

PUT &H20 <- [ス[°]-ス]を出力します。
 PUT &H41 <- [A]と出力
 PUT 65,66,67 <- [A][B][C]と出力

【PUT#】 機能：RS - 232C

書式 PUT# A1[,A2,A3]
 A1,A2,A3 : 変数 / 定数
 0 A1,A2,A3 &H7F

解説 A1, A2, A3の値をアスキーコードとしてCH1に出力

PUT# &H20 <- [ス[°]-ス]を出力します。
 PUT# &H41 <- [A]と出力
 PUT# 65,66,67 <- [A][B][C]と出力

```

10 I=0
20 PUT# &H41,&H42,&H43      ...[A][B][C]
30 PUT# &H44,&H45,&H4D      ...[D][E][CR]
40 I=I+1
50 AR(I)=GET#(0)           ...1文字入力
60 R1=RS(1)                ...CH1通信バッファのキャラクター数
70 IF R1<>0 THEN 40
80 FOR I1=1 TO I
90 PRINT AR(I1)
100 NEXT I1
RUN
65
66
67
68
69
13
    
```

(* このプログラムはコネクタ J 1 の 8 , 9 番ピンをショートさせなければ動作しません)

片仮名などの送信方法

MPC - 816はPRINTコマンドで片仮名などの8ビットキャラクターの送信が出来ません。そこでPUTコマンドで1キャラクターずつ出力します。

CNFG# 4,0,2 ノンパリ、データ8ビット、9600bps
 PUT# &H00B1,&H00B8,&H00BE "アケ"
 PUT# &H00D9,&H00D0 "llcr"

PUT # は使用前に CNFG # で RS - 232C の初期化を行って下さい。

【 PUTS # 】 機能：RS - 232C

書 式 PUTS# A1[,A2,A3]
A1,A2,A3 : 変数, 定数

解 説 PRINT # とほぼ同様ですが CR を出力しません。また、A1, A2, A3 が STR(n) であれば文字列出力となります。

PUTS# 123 456 789 .. "123【SP】456【SP】789" と出力し【CR】【LF】は出力しません。
PUTS# A B ... (A)【SP】(B) と出力します。(A) は変数 A の値の数字列です。(B) も同様
100 'AYAKO
110 PUTS# STR(100) ... ここでは "AYAKO" と出力されるのみです。

【 QUIT 】 機能：タスク操作

書 式 QUIT A1[,A2,A3]
A1,A2,A3 : タスクナンバー
1 A1,A2,A3 11

解 説 FORK されたプログラムの停止です。FORK されたプログラムが、END で終了していれば不要です。

MPG - 303 を使用中のタスクには直接 QUIT を実施しないで下さい。必ず STOP 4 や BSY(n) を併用して下さい。

【 REG 】 機能：MPG - 301

書 式 REG(reg)
reg : MPG アドレスと X3202 のレジスタ / カウンタセレクトコード
&Haaxx
aa : PG アドレス(省略時#1)
xx : セレクトコード
または -1 ~ -4 で動作状態ステータスレジスタ読み
-1 : MPG#1 ~ -4 : MPG#4

解 説 MPG - 301 に搭載されているパルス発生 IC「X3202」のレジスタのデータを読みみます。REG() は、1 または 2 byte のレジスタ読みです。3 byte 符号付きは読みは REG3() です。

MPG - 301 のコマンド / 関数 : CMND, REG(), REG3(), ST_REG

参照 : 「MPG - 301 詳細マニュアル」

WAIT REG(-1)=&H20 "MPG-301 #1 動作完了待ち

【 REG3 】 機能：MPG - 301

書 式 REG3(reg)
reg : MPG アドレスと X3202 のレジスタ / カウンタセレクトコード
&Haaxx
aa : MPG アドレス(省略時#1)
xx : セレクトコード

解 説 MPG - 301 に搭載されているパルス発生 IC「X3202」のレジスタのデータを読みみます。REG3() は 3 byte 符号付き読みです。

M P G - 3 0 1 のコマンド / 関数 : CMND, REG(), REG3(), ST_REG
参照 : 「M P G - 3 0 1 詳細マニュアル」

C=REG3(&H21) "MPG-301 #1 カウンタ読み

【 R E M 】 機能：編集

書 式 REM 文字列

解 説 コマンドはシングルクォート「'」で代用します。「R E M」と入力してもシングルクォートに置き変わります。尚、コメント文は S T R () 関数を文字列定数として使用できます。

```
100 'COMMENT
110 PRINT STR(-1)          <- "COMMENT" をターミナル画面に表示します
```

【 R E N U M 】 機能：編集

書 式 RENUM [n,o,s]

n : 新文番号

o : 旧文番号

s : ステップ量

0 n,o 32766

1 s 1000

解 説 RENUM <-最初から番号10番おきにふり直します。
RENUM 100 <-ふり直し開始番号が100となります。
RENUM 100,50 <-50番までの領域を除き、50番以降を開始番号100で10番おきにふり直します。
RENUM 100,50,5 <-これまで10番おきであったステップ量を5にすることができます。

【 R E T U R N 】 機能：制御文

書 式 RETURN

解 説 サブルーチン・コールは R E T U R N 文により元の実行ルーチンに戻ります。

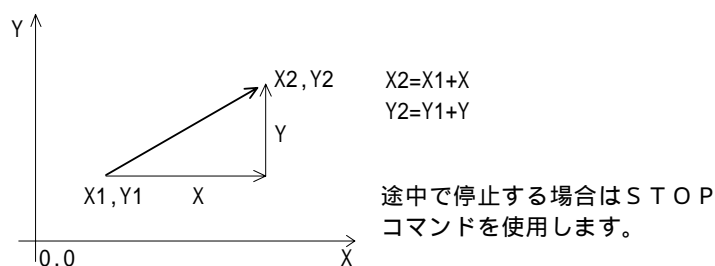
```
90 OFF 1
100 GOSUB 1000
110 ON 1 ←
120
|
1000 'AYA ←
1010 WAIT SW(1)=1
1020 RETURN
```

この様に、R E T U R N は G O S U B の次にジャンプします。G O S U B のネスト(多重コール)は、F O R 文と併せて 2 4 レベルまでです。

【 R M O V 】 機能：パルス

書 式 RMOV X,Y

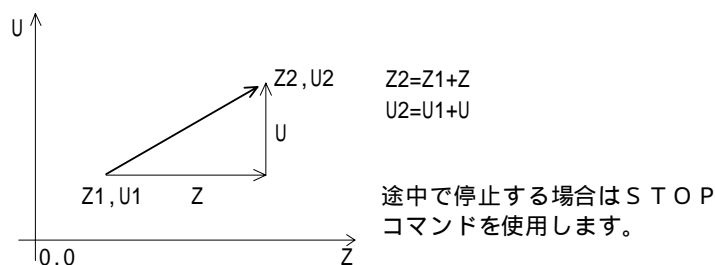
解 説 モード 5 , 6 とともにコマンド上の扱いは全く同等です。X Y 軸に対して X , Y パルス出力します。最大スピードは A C C E L、スピードの変更は F E E D コマンドによって行います。(直線補間)この時の現在位置を X 1 , Y 1 とするとパルス発生後の現在位置 X 2 , Y 2 は次の通りです。



【RMVZ】 機能：パルス

書式 RMVZ Z,U
 Z: Z軸移動量
 U: U軸移動量

解説 モード5, 6ともコマンド上の扱いは全く同等です。RMVZはRMOVコマンドがXY軸に対して有効なのに対してZU軸に有効となります。最大スピードはACCELによって定められスピードの変更はFEDZコマンドで行います。ZU軸に対してZ, Uパルス出力します。(直線補間)この時の現在位置をZ1, U1とするとパルス発生後の現在位置Z2, U2は次の通りです。



【RS】 機能：RS - 232C

書式 RS(n)
 n: RS - 232Cチャンネル番号 0または1

解説 RS - 232Cの受信キャラクタ数の読み取り。通信データが入力されていれば、その数を返します。
 RS(1): CH1(外部RS - 232C)です。
 RS(0): プログラム用ポートです。

PRINT RS(1) <-ポートのバッファに12個の文字データが入力されています。
 12

【RSV】 機能：タスク操作

書式 RSV(n)
 -1 n -128 (メモリーI/O)

解説 セマフォ関数と呼ばれるものです。複数のタスクで1個の装置を制御する場合にタスク間でインタロックをとる必要があります。この時にインタロックのテスト&セットをインタプリタで実施すると動作が保障されません。RSV(n)はメモリーI/Oに対してテスト&セットを実施します。

FORK 1,*MPG1
 FORK 2,*MPG2
 (略)

```

*MPG1
WAIT RSV(-1)=0
MOVE 0,0
OFF -1
GOTO *MPG1
*MPG2
WAIT RSV(-1)=0
MOVE 1000,1000
OFF -1
GOTO *MPG2

```

このプログラムでは2つのタスクが1つのMPG - 303に対してコマンドを出力しますがセマフォ関数RSV(-1)によってそれぞれのMODEコマンドが衝突することなく実行されます。セマフォの開放はOFFコマンドで行います。

【RUN】 機能：制御文

書式 RUN n
n：文番号(存在する文番号)
1 n 32766
またはラベル

解説 プログラム実行中は、キーボードからの入力はできません。TNYFSCのRUN命令は実行終了番号を指定できません。途中で実行を中止する場合は、BRKコマンドをプログラム中に挿入します。実行状態から元の初期状態に戻す為には、MPCのリセットボタンを押すか、キーボードから<CTRL>+<A>キーを押します。実行中に行番号の表示が必要であればTONします。TONの解除はTOFFとします。

【SET】 機能：パルス

書式 SET n,I,J
n：イン칭ング量バンク指定
0 n 3
I：X軸，Y軸，Z軸のイン칭ング量
J：U軸のJOG量
-32767 I,J 32767

解説 Tコマンド実行時(ティーチ・モード)では、0～3までのキーによってイン칭ング量を選択できる様になっていますが、その値はこのSETコマンドで設定しておきます。

```

>SET 0 10 10
>SET 1 20 20

```

【SETIO】 機能：I/O

書式 SETIO &HXX
XX：ヘキサ

解説 出力ポートを各バンク毎に論理設定します。各ビットは、次の様に各バンクに対応しています。負論理で1、正論理で0をセットします。

BIT 0	ポートバンク 0
BIT 1	" 1
BIT 2	" 2
.	.
.	.
.	.
BIT 6	" 6

負論理で1、正論理で0をセットします。MPCINITはSETIO &H7Fを実行します。シーケンサとして使用の場合は、負論理の為SETIO &H7Fとします。なお、SETIOだけで実行すると出力ポートを初期状態とします。MIO-248及びMIO-816#6~#11はこの設定から外れています。SETIOで設定された論理はMPCの電源を切っても、リセットボタンを押しても保持されます。

```
>SETIO &H7E          <-バンク0のトランジスタは全部ローインピーダンス
>ON 0                <-ポート0がハイインピーダンス
>ON 1                <- " 1 "
>ON 2                <- " 2 "
>SETIO              <-SETIOだけ実行するとともに&H7Fの状態に戻る。
```

【SETP】 機能：パルス

書式 SETP n, x, y
n: 点番号
x, y: 変数, 定数

解説 点nに座標x, yを入力します。点nへのX, Y座標値の設定

```
SETP 80,100,200      <-点(P(80))をx100, y100の値とします。
SETP 81,X(0),Y(0)   <-点(P(81))にx(0), Y(0)現在位置を入力します。
```

なお、nを0とすると現在位置(x, y)の意味となります。SETPOS x, yは、SETP 0, x, yとも記述できます。Z, Uに対してはSTPZUを用います。

```
SETP 80,100,200      <-点80のXを100, Yを200
STPZU 80,300,400    <-点80のZを300, Uを400
SETP 81,X(0),Y(0)   <-点81のX, Yを現在点
STPZU 81,Z(0),U(0) <-点81のZ, Uを現在点
SETP 0,0,0          <-現在点のX, Yを0,0
STPZU 0,0,0        <-現在点のZ, Uを0,0

X=100
Y=200
SETP 50,X,Y         <-変数でも可
SETP 100,X(100),0  <-点100のYだけ0
```

【SETPOS】 機能：パルス

書式 SETPOS X Y
X, Y: X軸, Y軸座標
-8388607 X, Y 8388607

解説 現在位置を設定します。原点復帰ではその位置を0, 0に設定しますが、これを変更する場合に用います。

```
HOME 0              <-HOME後の位置は0,0になる
SETPOS -10000,-10000 <-その位置を-10000,-10000に変更
```

この例では、X, Yともに原点を負に値としています。この様に原点を操作することにより実行エリアを増やすことができます。

【SETVAR】 機能：演算

書式 SETVAR V1,V2[,n]
V1,V2: 変数
n: 変数、定数

【SHMZ】 機能：パルス

書式 SHMZ u,z,s
u:U軸原点復帰方向
u=1(CW),2(CCW),0(無し)
z:Z軸原点復帰方向
z=4(CW),8(CCW),0(無し)
s:原点復帰スピード
1 s 32767
PG -1 の時 sはpps単位で指定 1 s 2000pps

解説 原点復帰(HOMZ)のz,u軸パルスの出力方向を定めます。

SHMZ 1,8,200	<-U: CW Z: CCW
SHMZ 2,4,100	<-U: CCW Z: CW
SHMZ 2,0,400	<-U: CCW Z: 原点復帰なし

【SHOM】 機能：パルス

書式 SHOM x,y,s
x: X軸原点復帰方向
x=1(CW),2(CCW),0(無し)
y: Y軸原点復帰方向
y=4(CW),8(CCW),0(無し)
s: 原点復帰スピード
1 s 32767
PG -1 の時 sはpps単位で指定 1 s 2000pps

解説 原点復帰(HOMZ)のx,y軸パルスの出力方向を定めます。

SHOM 1,8,200	<-X: CW Y: CCW
SHOM 2,0,400	<-X: CCW Y: 出力なし

尚、sと原点復帰時のパルスレートの関係は次の通りです。

モード5及びモード6

$$F=8 \times 10e6 / (413 + (52 \times s + 351) \times n)$$

nの値はSHRDによって変更されます。デフォルトは4です。sを10とすればF=2052.9となり2.05KHzとなります。つまりパワーオン後SHOM 2,8,10を実行しHOMEすれば約2Kのパルスレートで原点復帰します。約1Khzとするにはsを30とします。

【SHRD】 機能：パルス

書式 SHRD n
2 n 10

解説 HOME命令では、原パルスを出力する毎に原点入力をモニタし、HOMEコマンドで指定されたセンサパタンと比較します。この比較の回数を定めるのが、SHRDです。通常(リセット後)は、4と指定されていますが、サーボドライバなどのZ相(C相)検出ではパルス幅の狭いことや、エンコーダの振動などから検出が困難です。この為、SHRD 6~SHRD 8を指定し、検出を確実なものとし、nの値と原点復帰スピードの関係はSHOM参照

【SKIP#】 機能：RS - 232C

書式 SKIP# n
n: アスキーコード
0 n &H7F

解説 RS - 232C CH1のバッファにたまったnというASCIIコードを得るまで読み捨てます。指定された文字も読み捨てます。相手から出力された文字列が"A = 123"の場合"SKIP# &H3D" (&H3Dは"="です)を実行すると、"A ="という文字列が読み捨てられ、"123"という文字列が残ります。この後、【GETN#】を実行すると"123"という数値を得ることができます。

【SKPSP#】 機能：RS - 232C

書式 SKPSP#

解説 RS - 232C CH1のバッファにたまったスペース以外の文字に出会うまでスペースを読み捨てます。最初のスペース以外の文字がバッファの先頭にきます。相手から出力された文字列がスペースで区切られている場合に使用します。"x = 123"の場合SKIP# &H3Dを実行し、さらにSKPSP#を実行すると"123"の文字列のみが残ります。

【SP】 機能：ファイル

書式 点データの保存(セーブ)
SP n
n: ファイルメモリーナンバー
0 n 3

解説 点データをメモリエリアに保存します。保存できるエリアは4つ用意されておりますが、これらは配列エリアM(1200)~M(5999)と共通になっておりますので干渉しない様に使用して下さい。SPで保存したデータの読み出しはLPです。

```

10   FOR I=1 TO 10
20     J=I*100
30     K=I*1000
40     SETP I,J,K
50     NEXT I
>RUN
>PLS
P(1): 100 1000 0 0
P(2): 200 2000 0 0
P(3): 300 3000 0 0
P(4): 400 4000 0 0
P(5): 500 5000 0 0
P(6): 600 6000 0 0
P(7): 700 7000 0 0
P(8): 800 8000 0 0
P(9): 900 9000 0 0
P(10): 1000 10000 0 0
ok
>SP 1
>NEWP
>PLS

```

<-ポイントデータを作成します。
<-確認してみます。
<-ポイントデータを保存します。
<-メモリーのポイントデータをクリア
<-確認

```

P(1): 0 0 0 0
P(2): 0 0 0 0
P(3): 0 0 0 0
P(4): 0 0 0 0
P(5): 0 0 0 0
P(6): 0 0 0 0
P(7): 0 0 0 0
P(8): 0 0 0 0
P(9): 0 0 0 0
P(10): 0 0 0 0
ok
>LP 1
>PLS
P(1): 100 1000 0 0
P(2): 200 2000 0 0
P(3): 300 3000 0 0
P(4): 400 4000 0 0
P(5): 500 5000 0 0
P(6): 600 6000 0 0
P(7): 700 7000 0 0
P(8): 800 8000 0 0
P(9): 900 9000 0 0
P(10): 1000 10000 0 0
ok
>

```

消えている

<-保存したポインタを読み直す
<-確認

もとに戻った

【SQR】 機能：演算

書式 SQR(n)
n: 変数, 定数
0 n 32767

解説 nの平方根を求めます。

```

10 A=SQR(100)
20 PRINT A
30 END
RUN
10

```

nを負の数とすると平方根は負の数で出力されます。

【SQRT】 機能：演算

書式 SQRT n m v
n,m: 変数, 定数 >0
v: 結果が入る変数 (v=root(n*n+m*m))

解説 n、mの平方根を求めます。

```

>SQRT 3000 4000 V0
>PR V0
5000
>

```

【ST_REG】 機能：MPG - 301

書式 ST_REG reg data
reg: MPGアドレスとX3202のレジスタ/カウンタセレクトコード
&Haaxx
aa: MPGアドレス(省略時#1)
xx: セレクトコード
data: 設定データ

解説 MPG - 301に搭載されているパルス発生IC「X3202」のレジスタにデータを書き込みます。
MPG - 301のコマンド/関数: CMND,REG(),REG3(),ST_REG
参照:「MPG - 301 詳細マニュアル」

ST_REG 0,250 "周波数倍率
ST_REG 3,2000 "起動周波数

【STOP】 機能：パルス

書式 STOP n
n: 停止モード
n=1 減速停止
n=2 急停止
n=3 I/O指定
n=4 MPG - 303停止
n=5 MPG - 303停止解除

解説 n = 1, 2の場合
STOPコマンドは動作中のMPG - 303を強制停止させます。nが1の時は減速停止となり、nが2の時は急停止となります。

n = 3の場合
STOP 3,P,0
-128 P 512 ポート番号(#DEFS定義の文字列は不可,変数は可)
0=0,1 I/O状態 0=OFF,1=ON

STOP 3, P, 0ではポート番号及びON/OFFの指定を実施する事により直後の移動命令を減速停止することが出来ます。

100 STOP 3,16,1
110 MOVE 20000,20000

ポート16がONになったらMOVEが減速停止に入ります。STOP 3, P, 0は直後のコマンドに対してのみ有効です。

STOPコマンドは出力されたパルス数と座標値は整合されています。

```
>SETP 1 2000 0          <-P(1)を X=2000 Y=0に設定
>MOVE 0 0              <-X=0 Y=0に移動
>1000 MOVE P(1)        <-文番号1000を7x71で実行(点0,0からP(1)に移動します)
FORK 1 1000           <-急停止
>STOP 2               <-ティーチモード
>T                    <-急停止したところの座標
PG 1#1(X,Y,Z,U) 384 0 0 0 [XYZ,U] 800 800 <-ここからP(1)へ再移動
>MOVE P(1)            <-移動後の座標 P(1)
>T
PG 1#1(X,Y,Z,U) 2000 0 0 0 [XYZ,U] 800 800 <-移動後の座標 P(1)
>
```

STOPはHOME、PULSEコマンドにも有効です。

STOPコマンドによるパルス発生の途中停止の例です。(基本型)

```

MODE 5
PG 1
ACCEL 5000
FEED 0
PG 2
ACCEL 5000
FORK 1,*PG1          <-PG1はタスク1でFORK
FORK 4,*PG2          <-PG2はタスク2でFORK
INPUT A              <-スイッチの代わりの入力
PG 1                  <-PG1に切り替えパルス停止
STOP 1                <-減速停止
PG 2                  <-PG2に切り替えパルス停止
STOP 1                <-減速停止
WAIT BSY(1)<>0        <-PG1のパルスの停止を待つ      1
WAIT BSY(2)<>0        <-PG2のパルスも停止を待つ      1
QUIT 4,1              <-タスク1と4をQUIT
END
*PG1
MOVE 10000,10000
WAIT BSY(0)=1         <-正常停止でなければプログラムはここで停止する      2
TIME 10
MOVE 0,0
WAIT BSY(0)=1         <-正常停止でなければプログラムはここで停止する      2
TIME 10
GOTO *PG1
*PG2
MOVE 10000,10000
WAIT BSY(0)=1         <-正常停止でなければプログラムはここで停止する      2
TIME 10
MOVE 0,0
WAIT BSY(0)=1         <-正常停止でなければプログラムはここで停止する      2
TIME 10
GOTO *PG2

```

1タスク0でスイッチを監視してタスク1とタスク4でのパルス発生を停止するプログラムです。STOP 1 コマンドを実行してもすぐにパルスは停止しません。もしパルス発生中にタスクをQUITしてしまうとPGは異常な状態となります。STOPを実行した後はBSY()でパルスの停止を確認してタスクをQUITします。

2タスク0でSTOPコマンドを実行すると、実行されたPGがパルス発生中であればプログラムはそのステップから出て次の命令を実行します(ハングアップしない)。もしタイミング悪く次のパルス発生コマンド実行中にそのタスクがQUITされると減速停止されなくなるため、正常停止以外は次の命令を実行しない様にBSY()関数で監視します。

この様にSTOPをする方でもされる方でもパルスの状態を監視してうまくタスクを同期させなければなりません。そこでSTOP 1または2をする前にSTOP 4を実行するとMOVE後のBSY(n)監視が不要になります。

STOP 4、STOP 5について

STOP 4はコマンドが実行された後のMPG-303のパルス発生機能を停止させます。STOP 4がかけられたタスクがパルス発生中の場合は設定したパルス出力が終了してから次のパルス命令を受付なくなります。STOP 4はパルス発生コマンドを含むタスクのPAUSEやQUITを行う場合に便利なコマンドです。STOP 5はSTOP 4の解除です。STOP 4が解除されないままパルスコマンドを実行するとプログラムはそこでハングアップしてしまいます。

```

10 STOP 4
20 PG 1                <-PGコマンドは大丈夫だが
30 MODE 6              <-STOP 4がかけられたままだとハングアップする。

```

STOP 4、STOP 5の使用例 その

```

10 PG 1
20 FORK 1,*TASK1
30 *LOOP

```

```

40 INPUT A
50 STOP 4 <-MPG-303停止
60 WAIT BSY(0)<>0 <-パルス発生終了の確認
70 PAUSE 1 <-タスク-ス
80 '#1 PAUSE
90 PRINT STR(-1)
100 INPUT A
110 STOP 5
120 CONT 1
130 '#1 CONTINUE
140 PRINT STR(-1)
150 GOTO *LOOP
160 *TASK1
170 MOVE 0,0
180 MOVE 100000,0
190 GOTO *TASK1
>RUN
?
#1 PAUSE
?
#1 CONTINUE
?

```

STOP 4、STOP 5の使用例 その
パルス発生が連続しているタスクを停止させるにはSTOP 4でパルスコマンドをサスペンドさせてからSTOP 1またはSTOP 2を実行します。次の場合、もしSTOP 4をしないとSTOP 2で停止した直後に次のパルスコマンドを実行してしまい、パルスが停止しません。

```

PG 1
FORK 1,*PULSE
INPUT A
STOP 4 <-STOP 4でパルス発生コマンドをサスペンドします。
STOP 2 <-STOP 1またはSTOP 2でパルス発生を停止します。
WAIT BSY(1)<>0
QUIT 1
PRINT TASK(1)
STOP 5 <-STOP 4のサスペンドを解除します。忘れずに！
END
*PULSE
MOVE 0,0
MOVE 100000,100000
GOTO *PULSE

```

STOPコマンド使用上の注意

STOP、JOG、X(0)、Y(0)、BSY()などのパルス発生や現在点の取得の関数はMPG-303とのやりとりを行います。次のプログラムの様にこれらのコマンドや関数を連続して記述すると、パルスが停止しなくなるなど、正常に動作しないことがあります。

うまく動かない

```

100 JOG 100,1
110 WAIT SW(19)=1 ] JOGとSTOPの間にはWAITコマンドがあるので問題無い
120 STOP 1
130 Y=Y(0) ...STOPの後にすぐ現在点を取得している。これが良くない

```

└この他にも PRINT X(0),Y(0),U(0)やMOVE,RMOV,ACCEL,FEEDなどのMPG-303に関する命令

対策

```

100 JOG 100,1
110 WAIT SW(19)=1
120 STOP 1
125 TIME 10 ...少しタイマーを入れる。
130 Y=Y(0)

```

【STPZU】 機能：パルス

書 式 STPZU n,z,u
n: 点番号
z: z座標の値
u: u座標の値
1 n 300

解 説 点nへのz, u座標値の設定SETPと同様ですが、SETPはx, yの座標値の設定であるのに対し、STPZUはz, u座標値の設定となります。
P(1)を x:100 y:200 z:0 u:90と設定するには、

```
SETP 1,100,200  
STPZU 1,0,90
```

と2つのコマンドで設定することになります。尚、nを0とすると現在位置(z, u)設定の意味となります。

【STR】 機能：文字列

書 式 STR(n)
n: コメント文の文番号
1 n 32766 または -1 (-1は直前のコメント文を指定)

解 説 コメント文もしくはラベル文を出力する場合に使用。関数の値としてはコメント文の文字列の値を返します。nにはラベルは使用しないで数値のみ使用して下さい。与えられた文番号が無いと"VOID"を返します。また、STRで指定された値はRENUM実行時には編集の対象から除かれます。(使用する文字列を最初から10番おきに確保しておけばこの問題はありません。)この関数はPRINT文でも有効です。

```
100 'Data A?  
110 PRINT STR(100)
```

次のコマンドと関数により自由なフォーマットで通信することが可能となっています。例えば、次の様なフォーマットでデータを送りたい時は、次の例の様になります。

```
"a=1000 b-2000CR"  
1000 'a=  
1010 'b=  
1020 PUTS# STR(1000),1000  
1030 PUTS# STR(1010),2000  
1040 PUT# &H0D
```

次のサブルーチンは"AYAKO"を出力します。この様に-1を使用すると文番号に依存しないのでプログラムの見通しがよくなります。

```
1000 *AYAKO  
1010 PRINT# STR(-1)  
1020 RETURN
```

【SW】 機能：I / O

書式 SW(n)
 n: ポート番号
 0 n 255 I / O
 -128 n -1 メモリ-I / O

解説 SW(n)は、0または1の値を返します。SW(n)は、I F文、W A I T文等と組み合わせて使用する関数です。指定ポートがONであれば1、OFFであれば0を返します。

 WAIT SW(5)=1 <-ポート5がONになるまで待つこととなります。

 SW(n)は2回ポートを読みその値が確定している時に関数から抜け、その値を返します。SW(n)の論理反転として! SW(n)、2度読み無しのHSW(n)又、その論理反転として! HSW(n)があります。(MPC-816Kより5msecフィルターは無くなりました)

【!SW】 機能：I / O

書式 !SW(n)
 n: ポート番号
 0 n 255 I / O
 -128 n -1 メモリ-I / O

解説 !SW(n)はSW(n)と同等の関数ですが、指定ポートがONで0、OFFであれば1を返します。SW(n)の論理反転です。!SW(n)のタイマー無しは!HSW(n)です。

【T】 機能：パルス

TEACH参照

【TAIL】 機能：編集

書式 TAIL

解説 プログラム文番号の最大値を返します。編集中にプログラムを追加するとき、最終の文番号を知らない
と不便な場合があります。TAILによって得られた文番号より大きな文番号を与えてプログラムを追加
します。

【TASK】 機能：タスク操作

書式 TASK(n)
 n: タスク番号
 -1 n 11

解説 タスクの状態監視関数です。n = -1で自分のタスクのナンバーを得ることが出来ます。0 ~ 11の値
を与えると、対応するタスクの状態を与えます。状態値は次の通りです。

 0 : タスク実行中
 4 : タスク一時停止中 (P A U S E 状態)
 7 : タスク未使用 (Q U I T 状態)

 例えばコマンドで実行すると次のようになります。

```

>PRINT TASK(-1)          <-自分ほどのタスク?
0                        <-自分はタスク0
PRINT TASK(0) TASK(1)   <-タスク0と1の状態は?
0 7                      <-タスク0は実行中、タスク1は未使用
>

```

【TEACH】 機能：パルス

書 式 TEACH または、T

解 説 T N Y F S Cでは、4軸のデータを扱うことができます。点データP(n)は4次元のベクトル量となっており、ティーチング及びSETP, STPZUコマンドによって値を設定できます。ティーチ・モードへ移項する為に、TまたはTEACHとします。この時、現在位置表示、インチング移動量がそれぞれ表示されます。インチング量はコマンドSETによって0～3まで定めることができます。このモードは、キーを1つ押す度に所定の動作を実施します。

[0] インチング量0を選択します。

[1] インチング量1を選択します。

[2] インチング量2を選択します。

[3] インチング量3を選択します。

[X] + X インチング量だけ移動します。

[x] - x インチング量だけ移動します。

[Y] + Y インチング量だけ移動します。

[y] - y インチング量だけ移動します。

[U] + U インチング量だけ移動します。

[u] - u インチング量だけ移動します。

[Z] + Z インチング量だけ移動します。

[z] - z インチング量だけ移動します。

[O] ポートON

[F] ポートOFF

[P] ティーチング・モードで、“P”が押されると点番の入力モードとなります。改行があり、“P”表示されたら番号を入力すると、現在位置をティーチングします。

[Q] ティーチング・モード終了

[] スペース AXIS切り換え

[TAB] タブ PG1～3選択(MPG-303使用の場合)

インチング・・・キーを1回押すごとに一定量移動すること。

ティーチングモードにはターミナルソフトからしか入れません。

操作パネルのスイッチなどでポイントを設定したいときはJOGコマンドや、SETP、STPZUコマンドを使いプログラムします。

ティーチング例

ティーチングは最初は大きなJOG量で粗く、だんだん細かく動かして正確に位置を教えます。

TEACH <-TEACHまたはTでティーチモード

PGナンバ- JOG量(一回に動くパルス数)です。0,1,2,3の
TABキ-で切替 座標値 キ-で変わります。設定変更はSETコマンドです。

```
PG 1#1(X,Y,Z,U) 0 0 0 0 [XYZ,U] 400 400
Xキ-を1パルス
PG 1#1(X,Y,Z,U) 400 0 0 0 [XYZ,U] 400 400
Xキ-をもう1パルス
PG 1#1(X,Y,Z,U) 800 0 0 0 [XYZ,U] 400 400
Xキ-をさらに1パルス
PG 1#1(X,Y,Z,U) 1200 0 0 0 [XYZ,U] 400 400
Zキ-を1パルス
PG 1#1(X,Y,Z,U) 1200 0 400 0 [XYZ,U] 400 400
Zキ-をもう1パルス
PG 1#1(X,Y,Z,U) 1200 0 800 0 [XYZ,U] 400 400
Zキ-をさらに1パルス
PG 1#1(X,Y,Z,U) 1200 0 1200 0 [XYZ,U] 400 400
Zキ-をついでに1パルス
PG 1#1(X,Y,Z,U) 1200 0 1600 0 [XYZ,U] 400 400
Zキ-をおまけに1パルス
PG 1#1(X,Y,Z,U) 1200 0 2000 0 [XYZ,U] 400 400
```

P1 <-ここぞというところでPキ-を押しホィットナンバ-を入力(ここがP(1)になる)

```
PG 1#1(X,Y,Z,U) 1200 0 2000 0 [XYZ,U] 400 400
Xキ-を5パルス
PG 1#1(X,Y,Z,U) 3200 0 2000 0 [XYZ,U] 400 400
Zキ-を4パルス
PG 1#1(X,Y,Z,U) 3200 0 3600 0 [XYZ,U] 400 400
```

P2 <-ここぞというところでPキ-を押しホィットナンバ-を入力(ここがP(2)になる)

```
PG 1#1(X,Y,Z,U) 3200 0 3600 0 [XYZ,U] 400 400
```

Q <-ティーチングモードの終了はQキ-

PLS <-PLSコマンドでホィットの一覧を表示します。

```
P(1): 1200 0 2000 0
P(2): 3200 0 3600 0
P(3): 0 0 0 0
P(4): 0 0 0 0
P(5): 0 0 0 0
P(6): 0 0 0 0
P(7): 0 0 0 0
P(8): 0 0 0 0
P(9): 0 0 0 0
P(10): 0 0 0 0
ok
```

プログラム例

```
LIST
10 HOME 0
20 *LOOP
30 MOVE P(1) <-ティーチングしたP(1)に動きます
35 MOVZ P(1)
40 PRINT P(0) <-P(0),X(0),Y(0)など0を入れると現在の座標値を返します
50 TIME 10
60 MOVE P(2) <-ティーチングしたP(2)に動きます
65 MOVZ P(2)
70 PRINT P(0)
80 GOTO *LOOP
>RUN
1200 0 2000 0
3200 0 3600 0
1200 0 2000 0
3200 0 3600 0
1200 0 2000 0
```

【 THEN 】

IF参照

【TIME】 機能：タイマー

書式 TIME n
n: タイマー値
n=1で10msec
0 n 8388607

解説 時間待ちのタイマーで、10msec単位で設定
TIME 100 (100 × 10msec=1000msec=1秒)

【TMOUT】 機能：タイマー

書式 TMOUT n
n: タイマー値
n=1で10msec
0 n 8388607

解説 WS 1()、WS 0()のタイムアウト時間を設定します。TMOUTは、随時変更することができますが最後に実行された値が全てのWS 1()、WS 0()関数に適用されます。

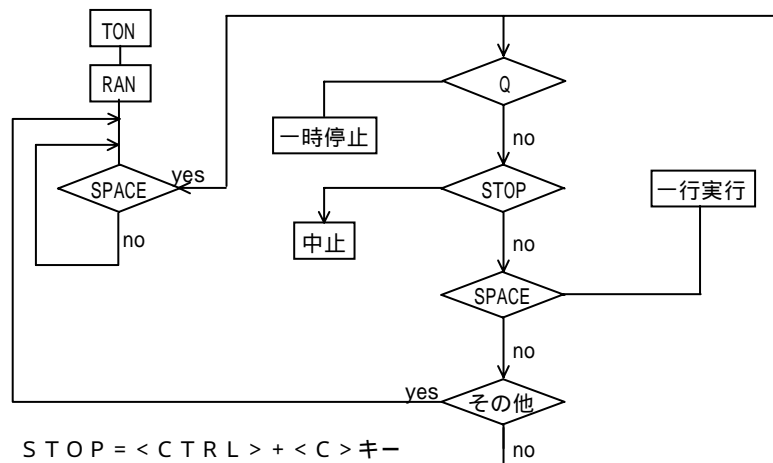
【TOFF】 機能：デバック

TON参照

【TON】 機能：デバック

書式 TOFF
TON

解説 メイン・タスクのみに適用。ダイレクトコマンドで実行します。トレース・モードは<SPACE>キー入力によって一時停止します。一時停止中には次の操作ができます。
<SPACE>キー入力で、1ステップ実行
<CTRL>+<C>キー入力で、中止
<Q>キー入力で、コマンドモードへ復帰
その他のキー入力で、継続
<Q>キー操作後はCNTコマンドで再実行、<CTRL>+<C>で中止します。尚、TONモードの解除はハード・リセットかTOFFを実行します。



【TST#】 機能：RS - 232C

書式 TST#(n)
 n: ダミー 0を設定して下さい。

解説 RS - 232C CH1の割り込み入力。バッファの先頭の文字がどのような種類のものであるか類別するものです。文字を取り出すことなく判別します。次の5種類です。

- 0: キャラクタ無し
- 1: 数字
- 2: 数字記号(+ / -)
- 3: その他の記号
- 4: 英字
- 5: 制御文字

例) バッファに "ABC" という文字列があって、"TST#(0)" を実行すると、

```
PRINT TST#(0)
4
```

となります。これは、先頭の文字 "A" が英字である為です。

【U】 機能：パルス

書式 U(n)
 n: 点番号
 0 n 300

解説 U座標を与えます。n = 0の時は、現在位置となります。

```
PRINT U(100)
90
```

P(100)のUの値が90であるということです。ティーチングプログラムでは、

```
STPZU 3,Z(0) U(0)
```

を実行すると、点3(P(3))に現在のZ及びUの値を設定することになります。
また、U(n)は配列として使用する事が出来ます。

```
U(n)=100
U(n)=U(3)+A1
```

【URANG】 機能：パルス

XRANG参照

【V_SWAP】 機能：メンテナンス

書式 V_SWAP

解説 P / Z版の切り替えをします。

```
TNYFSC(R) Rev-3.50d [VER-PmaX2044]   <-- P版
Copyright(C)by ACCEL CORP/BC-SOFT
[300p MPC-816K MPG MODE5]6]K8a9
>V_SWAP                               <--V_SWAP実行後はMPCの電源再投入。
TNYFSC(R) Rev-2.50b [VER-Zmx1729]   <-- Z版
```

Copyright (C) by ACCEL CORP/BC-SOFT
[255point MPC-816K MODE1~4]K8a9
>MPCINIT <--初期化
>ERASE
*

注意：出荷時はP版です。Z版はリピート・保守用として残されているものです。新規製作はP版でご使用下さい。

【 V E R 】 機能：メンテナンス

書 式 VER

解 説 バージョンには、バージョン・ナンバとそのリリース・デートが表示されます。バグレポート等の照合では、このバージョン表示によりどの問題を含むかをユーザーで判断して下さい。尚、VER 0とすればソフトの履歴を表示します。

【 V L I S T 】 機能：デバック

書 式 VLIST

解 説 A 0 ~ Z の変数の一覧表示です。

```
A0=9999
>A=12345
>VLIST
A0-9,A: 9999 0 0 0 0: 0 0 0 0 0: 12345
B0-9,B: 0 0 0 0 0: 0 0 0 0 0: 0
C0-9,C: 0 0 0 0 0: 0 0 0 0 0: 0
D0-9,D: 0 0 0 0 0: 0 0 0 0 0: 0
E0-9,E: 0 0 0 0 0: 0 0 0 0 0: 0
F0-9,F: 0 0 0 0 0: 0 0 0 0 0: 0
G0-9,G: 0 0 0 0 0: 0 0 0 0 0: 0
H0-9,H: 0 0 0 0 0: 0 0 0 0 0: 0
I0-9,I: 0 0 0 0 0: 0 0 0 0 0: 3600
J0-9,J: 0 0 0 0 0: 0 0 0 0 0: 1000
K0-9,K: 0 0 0 0 0: 0 0 0 0 0: 10000
L0-9,L: 0 0 0 0 0: 0 0 0 0 0: 0
ok <-Qで中断、その他のキ-で継続
>SETVAR A0,A,0
>VLIST
A0-9,A: 0 0 0 0 0: 0 0 0 0 0: 0
B0-9,B: 0 0 0 0 0: 0 0 0 0 0: 0
C0-9,C: 0 0 0 0 0: 0 0 0 0 0: 0
D0-9,D: 0 0 0 0 0: 0 0 0 0 0: 0
E0-9,E: 0 0 0 0 0: 0 0 0 0 0: 0
F0-9,F: 0 0 0 0 0: 0 0 0 0 0: 0
G0-9,G: 0 0 0 0 0: 0 0 0 0 0: 0
H0-9,H: 0 0 0 0 0: 0 0 0 0 0: 0
I0-9,I: 0 0 0 0 0: 0 0 0 0 0: 3600
J0-9,J: 0 0 0 0 0: 0 0 0 0 0: 1000
K0-9,K: 0 0 0 0 0: 0 0 0 0 0: 10000
L0-9,L: 0 0 0 0 0: 0 0 0 0 0: 0
ok
>
```

【 V P L S 】 機能：パルス

WPLS参照

【WAIT】 機能：制御文

書 式 WAIT ~条件式~

解 説 条件式が成立するまで停止します。

WAIT SW(n)=0 : SW(n)がOFFになるまで待つ

WAIT SW(n)=1 : SW(n)がONになるまで待つ

変数の条件待ちも可能な為、次の表現をとることができます。

WAIT A=1

この場合、他のタスクでAにセットしない限りそのタスクでは条件を待ち続けます。条件式は1 F文で使用されるものと同一

WAIT A<>B AとBが等しくない

WAIT A><B AとBが等しくない

WAIT A=B AとBが等しい

WAIT A>B AよりBが小さい

WAIT A<B AよりBが大きい

WAIT A=>B AよりBが小さいか等しい

WAIT A>=B AよりBが小さいか等しい

WAIT A=<B AよりBが大きい等しい

WAIT A<=B AよりBが大きい等しい

例)入力ポート1の信号に従って出力ポート2を制御しています。

```
100 #DEFS SEN1 1
110 #DEFO SOL1 2
120 *LOOP
130 WAIT SW(SEN1)=1
140 ON SOL1
150 WAIT SW(SEN1)=0
160 OFF SOL1
170 GOTO *LOOP
```

【WPLS】 機能：パルス

書 式 WPLS p r c

VPLS p r c

p : 出力ポート (out0かMIF - 816のパルスポート)

CW =1,4,16,64 (1=X,4=Y,16=U,64=Z)

CCW=2,8,32,128 (2=X,8=Y,32=U,128=Z)

r : オフパルス幅 2.17 μ 秒*r (次の変数がオンパルス幅 例えばrがB0ならB1がオンパルス)

c : パルスカウンター 省略可。いくつパルスを出したかを記録する変数

解 説 拡張パルス発生機能。WPLS, VPLSはPWMライクなコマンドです。周波数設定方法のほかは、YPLSと同様の使用方法になります。注意としてYPLS,WPLSはタイマー3,ZPLS、VPLSはタイマー2を共有しているため、それぞれ排他的にしか使用できません。

WPLS A,B0,C0

Aはポート指定です。YPLSと同様です。

B0はオフ時間を規定します。時間の単位は2.17 μ 秒です。50 - 65535(0.1m秒 - 0.14秒)まで指定できます。オン時間を規定するのは次のアドレスの変数です。この場合B1がオン時間指定変数です。50以下を指定するとパルスが正常でなくなったり、他のコマンドを受け付けなくなります。C0はカウンタの指定です。停止方法は停止待ちの方法はYPLSと同様です。

```
PLS_MIF
FORK 1,*TASK1
```

出力をMIFに変更

```

'=====
*LOOP0
'direction
A0=1                      方向 X-CW
'counter
C0=0                      カウンタークリア
'
FOR I=1 TO 10
'off pulse
B0=I*100                 オフパルス幅
'on pulse
B1=I*100                 オンパルス幅
WPLS A0,B0,C0
I1=I*1000
WAIT C0=I1
NEXT I
A0=0                      停止
WAIT A0=256              停止待ち
TIME 100
GOTO *LOOP0
'=====
*TASK1
'direction
A5=4                      方向 Y-CW
'off pulse
B5=200                   オフパルス幅
'on pulse
B6=200                   オンパルス幅
'counter
C5=0                      カウンタークリア
VPLS A5,B5,C5
WAIT C5=1000
A5=0                      停止
WAIT A5=256              停止待ち
TIME 50
'
A5=8                      方向 Y-CCW
C5=0                      カウンタークリア
VPLS A5,B5,C5
WAIT C5=-1000
A5=0                      停止
WAIT A5=256              停止待ち
TIME 50
GOTO *TASK1

```

PLS_MIF, YPLS参照

【WS0】 機能：I/O

書式 WS0(n)
WS1(n)
n: ポート番号
0 n 255 I/O
-128 n -1 メモリーI/O

解説 タイムアウト付き、ウェイト関数。インターロックとしては、次のコマンドと同様です。

```

WS0(n) : WAIT SW(n)=0
WS1(n) : WAIT SW(n)=1

```

WSは関数として用います。タイムアウト機能が追加されており、タイムアウトか条件成立で関数から抜けられます。WAITの条件が成立すれば0、タイムアウトならば1の値を返します。時間設定はTMOUTコマンドで行います。

```

10 TMOUT 500
20 IF WS1(-1)=1 THEN *TMOUT
25 'OK
27 PRINT STR(-1)
30 END
40 *TMOUT
50 'TMOUT
60 PRINT STR(-1)

```

```

70      END
>OFF -1
>PRINT SW(-1)
0
>RUN
OK
>ON -1
>PRINT SW(-1)
1
>RUN
OK
>

```

【WS1】 機能：I/O

WS0参照

【X】 機能：パルス

書式 X(n)

n : 点番号
0 n 300

解説 変数、定数の扱いと同様な為、演算に使用できます。n = 0の時は、現在位置となります。

```

>PRINT X(3) Y(3)
100,200

```

点3(P(3))のX, Yの値が100, 200であることを表示しています。

```

SETP 10,X(0),Y(0)

```

現在位置をP(10)のX, Yに設定します。また、X(n)は配列として使用する事が出来ます。

```

X(1)=100
X(2)=X(3)+A1

```

【XRANG】 機能：パルス

書式 XRANG max,min
YRANG max,min
URANG max,min
ZRANG max,min
-8388607 min<max 8388607

解説 XRANG ~ ZRANGは、JOGコマンドによる移動領域を制限するものです。指定した値に対してレンジで256パルスの誤差を持ちます。

```

XRANG 10000,0

```

と指定すると実際にはコマンドは9989の所で停止します。この計算方法は次の通りです。整数で演算します。

$$n' = (n/256) \times 256$$

これは、内部処理が最下位バイトを切り捨てて評価している為に発生します。

【 Y 】 機能：パルス

書 式 Y(n)
n : 点番号
0 n 300

解 説 変数、定数の扱いと同様な為、演算に使用できます。n = 0 の時は、現在位置となります。

```
>PRINT X(3) Y(3)  
100,200
```

点 3 (P (3)) の X , Y の値が 1 0 0 , 2 0 0 であることを表示しています。

```
>SETP 10,X(0),Y(0)
```

現在位置を P (1 0) の X , Y に設定します。また、Y (n) は配列として使用する事が出来ます。

```
Y(n)=100  
Y(n)=Y(3)+A1
```

【 Y P L S 】 機能：パルス

書 式 YPLS p,r,c
ZPLS p,r,c
p : 軸・方法指定
1:X-CW, 4:Y-CW, 16:U-CW, 64:Z-CW
2:X-CCW, 8:Y-CCW, 32:U-CCW, 128:Z-CCW
パルスは初期状態でMPC-816 J4 の出力ポートから出ます。
PLS_MIFでMIF-816 J5に切り換わります。
r : パルスレート 1 0 pps - 1 5 0 0 pps程度
c : パルスカウンター 省略可。いくつパルスを出したかを記録する変数

解 説 拡張パルス発生機能。引数はかならず、変数で指定します。コマンド実行前に第一、第二引数の値を決めておかないと正しく動作しません。第一引数、第二引数を動作中に変更するとパルス出力を停止、あるいはポート変更できます。また、第二引数に指定した変数を変更することによって、パルスレートも動的に変更できます。

Z P L S は Y P L S と同等のコマンドで Y P L S , Z P L S は同時に使用することができます。停止待ちはクリアしたポート変数が 2 5 6 になるのを待ちます。パルスはデューティー 5 0 % の正方形波です。

```
10 A=1  
20 B=100  
25 CO=0  
30 YPLS A,B,CO  
40 FOR I=1 TO 10  
42 IO=100*I  
45 PRINT IO  
50 WAIT CO=IO  
55 A=0  
60 WAIT A=256  
65 A=1  
70 YPLS A,B,CO  
80 NEXT I  
90 A=0  
>RUN  
100  
200  
300  
400  
500  
600  
700
```

停止
停止待ち


```
800
900
1000
>
```

PLS_MIF,WPLS参照

【Y R A N G】 機能：パルス

XRANG参照

【Z】 機能：パルス

書 式 Z(n)

n：点番号

0 n 300

解 説 変数、定数の扱いと同様な為、演算に使用できます。n = 0の時は、現在位置となります。

```
>PRINT Z(3)
100
```

点3(P(3))のUの値が1 0 0であることを表示しています。

```
SETP 10,Z(0),U(0)
```

現在位置をP(1 0)のZ , Uに設定します。また、Z(n)は配列として使用する事が出来ます。

```
Z(n)=100
Z(n)=Z(3)+A1
```

【Z P L S】 機能：パルス

YPLS参照

【Z R A N G】 機能：パルス

XRANG参照