第５章　コマンドリファレンス

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。

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・・・・・・①
B=A^&HF・・・・・②
```

①の例は、Aという変数に“1234”という値を代入しています。
②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。

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。

配　列

P版では、AR(31), X(300), Y(300), Z(300), U(300), M(5999)の6個の配列が使用可能です。Z版では、AR(31), X(256), Y(256), Z(256), U(256)の5種類です。

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、SW(), IN()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のパラレルデータとして読み取る。
```

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^C
論理和 （OR）	:	|	A=B|C
排他的論理和（WOR）	:	x	A=BxC

A=SW(0)^B,A=SW(0)|Bということも出来ます。

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。

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

コマンドリファレンスの書式

ON ←(1)

I/O ←(2)　　出力オン ←(3)

■書 式

ON A1 [A2 A3 A4 A5 A6]　←(4)

A1～A6:出力ポートナンバー

■解 説

・・・・・・・・・・・・　←(5)

関連:OFF,OUT　←(6)

(1) コマンド、関数の名前です。入力は大文字アルファベットで行います。
(2) コマンド、関数の分類です。
(3) コマンド、関数の機能です。
(4) 引き数の与え方を表します。[]は省略可能です。引き数が複数ある場合は1つ以上のスペースで区切ります。
(5) コマンド、関数についての内容や使用方法を解説しています。
(6) 関連するコマンド、関数です。

＊ラベル

制御文

■書 式
*文字列 文字列:半角英数字12文字以下

■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[Ent]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

制御文 **オンライン時の自動実行**

■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 秒以内になけ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

パルス **パルス出力の加減速設定**

■書 式

ACCEL n[,m,i]

n：最大スピード（pps）
　200≦n≦30000(MODE 5)
　2000≦n≦50000(MODE 6)

m：加速距離（pls）
　m≦5000 （省略するとm=n/10）

i：オフセットスピード（pps）
　0<i<1/2n

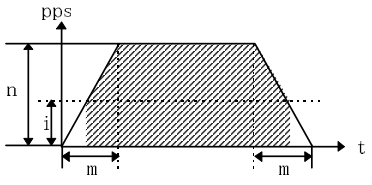
■解 説

n,mでFEED 0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4000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。

加速度 $= \frac{n^2}{2 \times m}\ \mathrm{pps/(sec)^2}$ この場合では、 $\frac{(4000)^2}{2 \times 400} = 20\mathrm{k/(sec)^2}$ 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、 $\frac{2 \times 400}{4000} = 0.2\,\mathrm{sec}$ 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、あるパルスレート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。



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

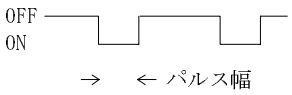
ACCELコマンドはRAM上に加減速テーブルを作成しますが、演算に若干時間がかかります。時間は指定されたパラメーターにより変わり、緩やかな加減速ほど時間がかかります。ACCELコマンドはプログラムの先頭で行い途中でMOVEや、JUMPのスピードを変えるにはFEED、FEDZなどを使用した方がタイムロスが少なくて済みます。

```
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				
	MODE5		MODE6	
n	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
ﾊﾟﾙｽ幅	20μsec		4μsec	

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。

①一般的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
```

②MODE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     <-この程度ならアっというまに終わる。
100       GOTO 40
```

③RMOVコマンドを使って１パルスずつ出力する。

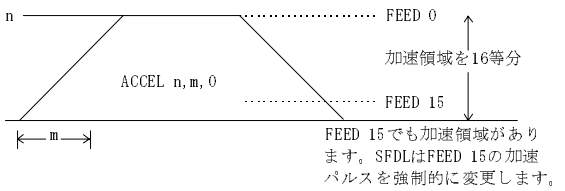
RMOVの後にタイマーを用いればどんなに遅いパルスも出力できます。

```
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
```

④MODE、ACCEL設定を変えずにスピードを落とす方法としてSFDL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      SFDL 1
90      FEED 15
100     GOTO 40
```



⑤PULSE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。

※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。

MPG-303初期設定順序例

```
PG 1                    ″必ず実行してください。
MODE 5
ACCEL 3000 300 100
FEED 0
D45 0
OVRUN 0
```

AD

MIF-816AD　　AD変換

■書 式

AD(n)

n：入力チャンネル番号

0≦n≦3

■解 説

MIF-816AD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J4の28,29,30に対応します。返される数値は、0～4095(分解能12bit)で1mV単位です。

```
100 V=AD(0)
110 PRINT V
```

ch3はDAのループバックです。複数のタスクから同時に実行すると、正しいデータが読めません。排他的になるようにプログラムでセマフォをかけて下さい。

DA参照

AR

演算 **配列変数**

■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。AR<ent>とすると配列の内容を表示します。

```
AR(0)=|N(0)
AR(1)=5
PR|NT AR(0)
```

AR(n)はSFTR、SFTLコマンドでデータのシフトができます。

配列変数AR(n)とSFTRの使用例

```
10 FOR |=0 TO 7
20 AR(|)=|                    ″配列変数AR(0)～AR(7)にデータを入れます。
30 NEXT |
40 '
50 *LOOP
60 PR|NT AR(0),AR(1),AR(2)    ″AR(0)～AR(7)の内容を表示
70 PR|NT AR(3),AR(4),AR(5)
80 PR|NT AR(6),AR(7)
90 PR|NT
100 SFTR                      ″右方向へ1回シフトします。
110 AR(0)=AR(8)               ″はみ出たAR(8)(元AR(7))をAR(0)に代入
120 T|ME 50
130 GOTO *LOOP
140 '
>RUN
0 1 2   ┐
3 4 5   │①
6 7     ┘
7 0 1   ┐
2 3 4   │②
5 6     ┘
6 7 0   ┐
1 2 3   │③
4 5     ┘
5 6 7   ┐
0 1 2   │④
3 4     ┘
```

SFTR,SFTL参照

ATAN

演算 **アークタンジェント**

■書 式

ATAN n m v

n,m:変数,定数 >0

v:結果が入る変数

■解 説

n、mのアークタンジェントを求めます。 (v=atan(m/n))

```
ATAN 10000 10000 A
>PR A
4500
>ATAN 10000 17321 B0
>PR B0
6000
```

45° を超えたら引数を反対にして補角を計算

AXIS

パルス **単軸パルス出力ポート選択**

■書 式

AXIS n

n：軸番号

1≦n≦4 1:X 2:Y 3:U 4:Z

■解 説

PULSEコマンドなどの単軸のパルス発生の軸を選択します。

```
10 PG 1
20 *LOOP
30 AXIS 1              "X軸からﾊﾟﾙｽ出力
40 PULSE 1000
50 AXIS 2              "Y軸からﾊﾟﾙｽ出力
60 PULSE 1000
70 AXIS 3              "U軸からﾊﾟﾙｽ出力
80 PULSE 1000
90 AXIS 4              "Z軸からﾊﾟﾙｽ出力
100 PULSE 1000
110 GOTO *LOOP
```

ティーチモードでのAXIS選択

```
PG 1#1(X,Y,Z,U) 0 0 0 0 [XYZ,U] 5 5
   ↑ ↑
   |  └AXIS 選択:スペースキー
   └PGボード選択:TABキー
```

PULSE参照

BKCNT

デバッグ　　BRKコマンドのパスカウント

■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 I
100 FOR I=1 TO 30
110 ON 1
120 TIME 100
130 OFF 1
135 BRK                    'BRKを挿入してデバックする。
140 IF SW(1)=1 THEN 160
150 ON 2
160 NEXT I
BKCNT 13
RUN
```

前記の様にBKCNTを設定し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。

BRK参照

BL_AND

I/O　　条件出力･論理積ビットロード

■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

■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

I/O　　条件出力･論理和ビットロード

■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 SET I O
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、
                            ″両方がOFFならば出力0はOFF
30 PR I NT 0_SW(0)
40 T I ME 5
50 GOTO 20
>RUN
 0
 0
 1
 1
```

IF文で書くとこんな感じ．．．

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

BRK

デバック　　プログラムの一時停止

■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。
※メインタスクだけにしか使えません。

CNT参照

BSY

パルス　　MPG-303の動作状態関数

■書 式
BSY(n)
n:PG番号 0≦n≦3
0 現在アクセス中のPG
1～3 PG番号

■解 説
MPG-303の動作状態を数値で返します。引数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または3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をかえします。

STOP参照

CMND

MPG-301　　X3202コマンド実行

■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。

参照 :「MPG-301 製品別マニュアル」

```
CMND &H08          ″定速連続駆動=>スピードは起動周波数(REG 3)
CMND &H0100        ″インデックス駆動(+)
CMND &H0131        ″減速停止
```

極定速モード　SP4=OPEN SP5=SHORT　U5に専用PLD実装する

```
CMND -1            ″極定速モードに移行
CMND -2            ″極定速モード解除
```

この設定は全部MPG-301に有効

REG(),REG3(),ST REG参照

CNFG#

RS-232C　　外部RS-232Cの初期化

■書 式

CNFG# A1,A2,A3

A1:モード設定 0≦A1≦7

0:NP7B1S 1:NP7B2S 2:P7B1S

3:P7B2S 4:NP8B1S 5:NP8B2S

6:P8B1S 7:P8B2S

A2:パリティ選択 0≦A2≦3

0:遇パリティ XON/XOFF 有

1:奇パリティ XON/XOFF 有

2:遇パリティ XON/XOFF 無

3:奇パリティ XON/XOFF 無

A3:ボーレート選択 1≦A3≦6

0:禁止 1:19.2K 2:9600

3:4800 4:2400 5:1200

6:600 7:禁止

■解 説

外部RS-232Cポートの設定を行います。MPC-816のJ1には、コマンド制御可能なシリアルポートSG(7),TX1(8),RX1(9)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。

※19.2kは安定動作しないことがあります。

```
CNFG# 4,2,2
  ″ノンパリ データ8ビット 1ストップビット
  ″XON/OFF無し
  ″9600bps
```

A1～A3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 CH0 : 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

デバッグ　　プログラムの再実行

■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。

BRK参照

CONT

タスク操作　　休止プログラムの再実行

■書 式
CONT n[,m,l]
 n,m,l:タスク番号
  P版 1≦n,m,l≦11
  Z版 1≦n,m,l≦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。

PAUSE,FORK,QUIT参照

CSW

I/O　　エッジ検出

■書 式
CSW(n)
 n:入力ポート番号
  0≦n≦255　I/O
  -128≦n≦-1 メモリ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。
A=CSW(n)では、 nがOFF->ONのとき A=1、 nがON->OFFのとき A=0。

```
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

I/O　　エッジ検出/論理反転

■書 式
!CSW(n)
 n:入力ポート番号
  0≦n≦255　I/O
  -128≦n≦-1 メモリ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CSW(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。
A=CSW(n)では、 nがOFF->ONのとき A=0、 nがON->OFFのとき A=1。

D45

パルス　　移動モードの設定

■書 式

D45 n

n=0:XY,ZUいずれも直線補完

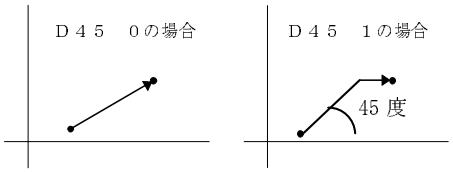
n=1:XY,ZUいずれもD45移動

n=2:XYのみD45移動、 ZUは直線補完

n=3:ZUのみD45移動、 XYは直線補完

■解 説

MPCのパルス発生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XY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。 D45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。
MPC起動後のnは不定です。 プログラムで初期化して下さい。



D45 0の場合　　XY同時スタート・ストップ

D45 1の場合　　45° 移動(XY同一パルスレート)後単軸移動

DA

MIF-816AD　　DA出力

■書 式

DA n

0≦n≦4095

(分解能12bit 0~4.095V(1mV/bit))

■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。出力は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

AD参照

DEC

制御文　　繰り返し制御文

■書 式
FOR a1=a2 TO a3
DEC a1
 a1:変数
 a2,a3:変数、 定数 a2>a3

■解 説
IF THEN,GOTOで飛びだし不可

#DEFO

I/O　　出力ポートのシンボル定義

■書 式
#DEFO 文字列 番号
 文字列:8文字以下
 番号:32767以下

■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、この定義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。
※MBK-816のエリア(40000台、 50000台)は定義できません。

#DEFS参照

#DEFS

I/O　　入力ポートのシンボル定義

■書 式
#DEFS 文字列 番号
 文字列:8文字以下
 番号:32767以下

■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
```

定義済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                  ' はじめのプログラム
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)
>
```

※MBK-816のエリア(40000台、 50000台)は定義できません。

#DEFO参照

DEF_RST

I/O　　リセット入力の指定

■書 式

DEF_RST A

A=&H00～&HFF 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
```

DELETE

編集　　ブロック単位で削除

■書 式
DELETE n,m
n,m: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までを削除します。ある行より後を全て削除したい場合には、次の様にすると有効です。

```
32766 END
DELETE 100 32766
```

この様に32766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TAILコマンドで最終行を調べ、これにより全消去を実施しています。

#DI

タスク操作　　マルチタスクの停止

■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入れないことです。

#EI参照

E

RS-232C　　CH1のエラー検出

■書 式
E(n)
n=0

■解 説
P版ではRS-232C 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: パリティーエラー
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-232C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。

#EI

タスク操作　　マルチタスクの再開

■書 式
#EI

■解 説
#DIによって停止されていたマルチタスクを再開します。

#DI参照

ELSE_GOSUB

制御文　　IF文でのELSE制御

■書 式
ELSE_GOSUB n
　nはラベルまたは文番号

■解 説
IF文条件結果はその直後の行まで保存されています。(GOSUB中の実行によっても破壊されません)また、GOSUBによってRETURN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*JOB1か*JOB2のいずれかが実行され、*JOB1を実行した場合は*JOB2が実行されることはありません。

IF参照

ELSE_THEN

制御文　　IF文でのELSE制御

■書 式
ELSE_THEN n
　nはラベルまたは文番号

IF参照

END

制御文　　プログラム終了

■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。END行がみつからないと下の行を次々と実行します。END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END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
```

ERASE

メンテナンス　　MPC-816初期化

■書 式
ERASE

■解 説
フラッシュROM(FROM)のプログラムをクリアーします。プログラム編集後のRUN、またはFWRITEによりFROM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「MPCINIT」は FROM のプログラムはクリアーしません。MPCINITをしてSRAMのプログラムを消去しても、次のパワーオン時にFROM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 ********      ″FROMに書き込み
>MPCINIT                                ″SRAMの初期化
>ERASE                                  ″FROMプログラムクリアー
*                                       ″ERASEマーク
>
```

MPCINIT参照

ERR_ON

I/O　　プログラム停止時のエラー表示

■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と、 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。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。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
```

FEDD

パルス　　ゲートモーション下降スピード

■書 式
FEDD 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、上昇スピードより遅く動作する必要があります。そのため、下降時のみ別にスピード設定ができます。尚、FEDDはFEDZより優先順位が低い為FEDZの後に実行されます。FEDZはFEDDの値を自分の値と同一に設定します。これは、FEDDの設定忘れを避ける為です。

FEDH

パルス　　HOME時のスピード設定

■書 式
FEDH n
 0≦n≦15

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

FEDZ

パルス　　Z軸のスピード設定

■書 式
FEDZ n
 0≦n≦15 (PG 1～3)
 0≦n≦63 (PG -1)

■解 説
U、Z軸のスピードを選択します。

FEED参照

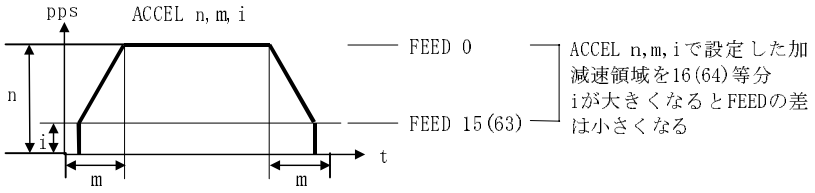
FEED

パルス　　MOVE、RMOVのスピード設定

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

■解 説
ACCEL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ACCELによって最大使用レートを定め、プログラム実行中にスピードを変更する場合にFEED等を用います。FEEDはACCELと異なり瞬時に作業を終了します。nで指定されるスピードについてはACCELコマンドを参照して下さい。

FEED　X,Y軸のスピードを定めます。
FEDZ　U,Z軸のスピードを定めます。
FEDD　JUMPでの下降移動のスピードを定めます。FEDDはFEDZの後に指定します。
FEDH　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      ″ACCELコマンドは時間がかかる
SETPOS 0,0
*LOOP
FOR I=0 TO 15
FEED I                     ″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ROMV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しないとスピードはFEED 0
>HOME 0
>RMOV 20000,0          ″4.3sec HOMEするとスピードはFEED 8
>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
```

ACCEL,PG参照

FIND#

RS-232C　　文字の検索/RS-232C CH1

■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(I)=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

制御文　　繰り返しループ

■書 式

加算

FOR a1=a2 TO a3

NEXT a1

a1:変数

a2,a3:変数、 定数 a2<a3

減算

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
 (略)
 149
 150
 # 30
!! Out of Range
```

引き数にX(n)等の配列は使えません。変数・定数にして与えてください。①のようなタイプはエラーにはなりませんが無効になります。②の場合は、動作しました。

①

```
10      FOR X(1)=1 TO 10
20       PRINT X(1)
30      NEXT X(1)
```

②

```
5        X(1)=5
10      FOR I=1 TO X(1)
20       PRINT I
30      NEXT I
```

FORK

タスク操作 **タスクの起動**

■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事ができます。

```
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                          ″タスクポーズ
 ?
 7                          ″タスク消滅
```

※すでにFORKされているタスクをまたFORK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                          ″FORKでタスク1は最初に戻る
 1
 2
```

QUIT,CONT,PAUSE参照

FREE

編集 **プログラム残量表示**

■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                       ″行数
```

FWRITE

メンテナンス **フラッシュ ROM書込**

■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
>
```

RUN参照

GET#

RS-232C　　1文字入力関数/CH1

■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。 入力があるまで待ちます。
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。

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

このプログラムは、 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(X(n)を配列変数として扱う)場合は

```
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トダウンや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-816のRS-232Cの受信については配慮をして使用するということでお願いします。

```
●MASTER
*AG
 A=49
 CNFG# 4,0,2
 PUT# A
 C=10
 5-23
*LP
```

```
●SLAVE
*AG
 CNFG# 4,0,2
 A=GET#(0)
 TIME 15
 C=10
*LP
 PUT# A
```

```
 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
```

```
 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
 5-24
 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)
 IF A<>A8 THEN *RSE
 RETURN
*RSE
 PRINT STR(-1)
 PRINT T1,RS(1),A8
 END
```

GET

RS-232C　　1文字入力関数/CH0

■書 式

GET(0)

■解 説

RS-232C CH0(プログラムポート)に適応される他はGET#(0)と全く同等機能です。

GETN#

RS-232C　　数字列の読み込み/CH1

■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を処理を終了します。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#     ″┐GET#の割込み禁止により
30 GETN# B0   ″┘データがバッファに入らないことがある
40 SKIP# &H0D
```

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#         ″┐Gバッファの中のデータを読むので
50 GETN# B0       ″┘割り込み禁止の影響なし
60 SKIP# &H0D
```

INPUT#参照

GOSUB

制御文　　サブルーチンコール

■書 式
GOSUB n
 n: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
```

RETURN参照

GOTO

制御文　　指定の文番号へジャンプ

■書 式
GOTO n
n: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。
※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)

IF,GOSUB参照

HCSW

I/O　　CSW(n)の高速読み取り

■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

パルス　　原点へ移動

■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

パルス　　XY原点復帰

■書 式

HOME n,x,y

n:XS1～YS2までのセンターパターン

x,y: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。

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

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。

```
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。

```
10 MODE 5              ″MODE設定は一番始め
20 OVRUN 0             ″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        ″X軸CW方向,Y軸は無し,速度353pps
50 FEDH 15             ″退避移動速度
60 HOME 1,-1000,0      ″原点復帰 XS1がONまで,退避量X=1000ﾊﾟﾙｽCCW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置として残ります。

```
>HOME &HF 500 500      ″退避量500パルスでX,Y軸の原点復帰
                       ″<CTRL>+<A>で停止する
       TASK 0 # 30
T                      ″TEACH MODEで見ると...
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をタスク1で実行
>STOP 1                ″タスク0から強制停止
>T
PG 1#1(X,Y,Z,U) 0 0 0 0 [XYZ,U] 400 400
```

■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、そのうちにどの様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
>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。
                  ″ボードを交換したり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))

SHOM参照

HOMZ

パルス　　ZU原点復帰

■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
HOMZコマンドの実行後の座標値
①U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     ″TEACHモードで確認
>SHMZ 1,0,1000         ″原点復帰 U:CW Z:無し スピード:1000
>HOMZ 1,0,-20          ″原点復帰 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    ″TEACHモードで実行後の座標確認
```

②Z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 U:無し Z:CW スピード:1000
>HOMZ 4,-200,0         ″原点復帰 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   ″TEACHモードで実行後の座標確認
```

③U,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 U:CW Z:CW スピード:1000
>HOMZ 5,-200,-200      ″原点復帰 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    ″TEACHモードで実行後の座標確認
```

これら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。

SHMZ参照

HPT

I/O　　MPG-303原点入力の読み込み

■書 式

HPT(n)

0≦n≦8

n=0:パラレル入力

n=1~8:ビット入力

■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事ができます。

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ON、XS2がOFFの状態です。HPT(n)は実行されたタスクに対応するMPG-303の原点入力を返しますが、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ーロックをとって下さい。

HSW

I/O　　SW(n)の高速読みとり

■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

■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の多用は結果的にタクス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して下さい。

SW,!HSW参照

!HSW

I/O　　HSW(n)の論理反転

■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

■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。

SW,HSW参照

HWS0

I/O　　入力タイミング待ち関数

■書 式
HWS0(n)
HWS1(n)
 n : ポートナンバー
  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(0)=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、タイムアウトとなれば1の値を返します。タイムアウトの時間(制限時間)はTMOUT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。

TMOUT参照

HWS1

I/O　　入力タイミング待ち関数

■書 式
HWS1(n)
 n:ポートナンバー
  0≦n≦255 I/O
  -128≦n≦-1 メモリー I/O

HWS0,TMOUT参照

IF

制御文 **条件分岐**

■書 式

IF～条件式～THEN n
［ELSE_THEN n］
IF～条件式～GOSUB n
［ELSE_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が大きいか等しい
IF A<=B THEN/GOSUB AよりBが大きいか等しい

IN

I/O **パラレル入力**

■書 式

IN(n)
n:バンクナンバー
0≦n≦31 I/O
-16≦n≦-1 メモリー I/O

■解 説

入力ポートのパラレル入力。IN(n)には5msecのフィルターはありません。SW(n)とIN(n)の関係は次の通りです。

MPC-816	入力(SW(n))	IN(n)
	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DSW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DSW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
```

SW参照

INPUT

RS-232C　　CH0からの数値入力

■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
30 END
RUN
?123    (123はキーボードにより入力)
123
```

INPUT#

RS-232C　　CH1からの数値入力

■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・ターミネーターに限られ次の書式となります。 (デリミタは「スペース」(&H20)、 ターミネーターは「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 *AH0
90 NEXT A
100 GOTO *LOOP
110 *AH0
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。

```
INPUT# A1     ″ ←&H31&H32&H33&H0Dと送るとA1に123と入る。
              ″ &H30~&H39以外のキャラクタが混ざると不正データとなる。
              ″ 10進数値だけ取るにはGETN#。
```

GET#,GETN#参照

IO_CNT

ソフトカウンタ　　カウント

■書　式

IO_CNT

■解　説

MPC-816の入力を使った3byte長のカウンタで、3チャンネルありバックグランドで動作します。in0,2,4をトリガ入力、in1,3,5を方向指示として使用し、変数C0,C2,C4をカウンターとして使用します。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(500pps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)　onでC0が変化
　　in1 (up/down) offで正/onで負
[カウンタ1]　in2 (count)　onでC2が変化
　　in3 (up/down) offで正/onで負
[カウンタ2]　in4 (count)　onでC4が変化
　　in5 (up/down) offで正/onで負

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

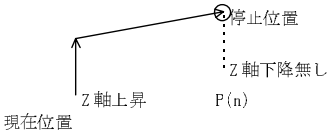
OUT CNT,OUT CSET参照

JMPZ

パルス　　ロボット移動命令

■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。これは調整中の便をはかる為設定されたコマンドです。



JOG

パルス　　ジョグ移動

■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

■解 説
JOGコマンドは、PGコマンドの中で唯一パルス発生中に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XRANG,YRANG,ZRANG,URANGコマンドで設定されます。JOGコマンドでは各RANGの上位2byteのみを比較して下位1byte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1byteを切り捨てると言うことはうちわで255パルスの範囲でRANGが設定されます。次のプログラムの様にXRANGを1000(&H3E8)に設定しても下位1byteが切り捨てられるので実際には768パルス(&H300)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。
                        ″ XRANG 1024,-1025と設定すると+1024,-1025まで動きます。
50 *LOOP
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
```

スピードnの最大値はACCELコマンドの加速距離と関連します。

```
10 PG 1
20 MODE 5
30 ACCEL 30000,900,500    ″加速距離900パルス
40 JOG 1000,2             ″スピード > 加速距離なのでNG
RUN
 # 40
!! Out of Range          ″エラーになる
>
```

上の30行を

```
30 ACCEL 30000,1000,500    ″加速距離変更
40 JOG 1000,2              ″スピード≦加速距離 でOK
```

■動作範囲のMINが0の場合

次のプログラム①の様に、XRANGの最小側(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。

①

```
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  <-------------------------┘
```

②

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

③

```
        |
50     XRANG 10000,-256
        |
>RUN
 -1
```

④

```
        |
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     ″この場合JOGの加速距離は2000まで
>JOG 2000,1
>STOP 1
>JOG 2001,1
!! Out of Range
>

ACCEL 10000                ″mを省略するとnの1/5まで設定可能
>JOG 2000,1
>STOP 1
>JOG 2001,1
!! Out of Range
```

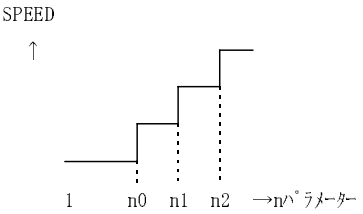
```
>ACCEL 20000
>JOG 4000,1
STOP 1
>JOG 4001,1
!! Out of Range

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

＊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
|WATSU SC-7201 UN|VERSAL 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。



MODE 5

ACCEL の設定値	MOVE,RMOV ｺﾏﾝﾄﾞ MAX(KHz)	JOG ｺﾏﾝﾄﾞ	
		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()内はﾊﾟﾗﾒｰﾀｰ設定値。最高 ACCEL の 1/5 以下で最大 5000 まで。

MODE 6

ACCEL の設定値	MOVE,RMOV ｺﾏﾝﾄﾞ MAX(KHz)	JOG ｺﾏﾝﾄﾞ	
		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()内はﾊﾟﾗﾒｰﾀｰ設定値。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 のﾊﾟﾗﾒｰﾀｰに関わらず 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±32767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。
```

※PG-1ではJOGは使えません。JOG代用はYPLS。

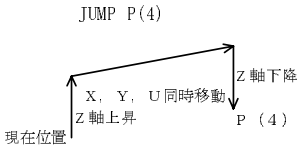
XRANG,YRANG,URANG,ZRANG参照

JUMP

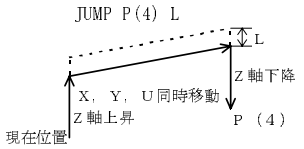
パルス　　ゲートモーション

■書 式
JUMP P(n)[,L]
 n:点番号 1≦n≦300
 L:Z軸の上限値 0≦L≦8388607
  L=0なら最上端まで上昇

■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。



①JUMP P(4)



②JUMP P(4) L

前記の図はゲートモーションを表現しています。①は上限指定がない為Z軸が0まで戻っています。②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 MODE 1 ではU軸をあわせた3軸同時移動となります。

KEY

メンテナンス **フラッシュ ROMロック**

■書 式

KEY n
 n:ロック番号

■解 説

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。 パスワートは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表示もしなくなります。

```
KEY -123     ″負のキー
*+++++++++   ″ロック中
LIST
--Hided--    ″見えません
```

※番号は忘れないように

LET

演算 **演算子**

■書 式

LET 算術式
（LETは記述しても自動的に省略されます）

■解 説

TNYFSCでサポートされているのは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はスモールエックス）

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。
P版では3byte整数、Z版では2byte整数の演算となります。

LIST

編集　　プログラムリスト表示

■書 式
LIST [n,m]
 n:開始文番号
  0≦n≦32766 またはラベル
 m:表示行数
  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。

```
L|ST                 ″前回の続きを表示
L|ST 0               ″先頭から表示
L|ST 30              ″文番号30から表示
L|ST 30,5            ″文番号30から5行表示
L|ST *MA|N           ″ラベル*MA|Nから表示
```

LSCNT参照

LOC

LCD　　カーソル位置指定/初期化

■書 式
LOC n,m[,s]
 n:行(縦) 1～2:(2行CLD)
  1～4:(4行CLD)
  0:LCD選択・消去(下表)
 m:列(横) 1～16:(16列LCD)
  1～20:(20列LCD)
  (n=0のときは下表)
 s:カーソル
  s=15:カーソル+ブリンク
  s=14:カーソル
  s=13:ブリンク
  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。

CLD機種選択(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

```
10  LOC 0,2,12              ″初期化:    BUS接続2*16
20  LOC 1,1                 ″表示位置:   1行1列
30  PRC &H41,&H42,&H43      ″キャラクタ表示: ABC
40  LOC 2,5                 ″表示位置:   2行5列
50  ' ACCEL
60  PRS 5                   ″文字表示: 上行のコメント文字列(5文字)
70  FOR I=0 TO 999
```

```
80   LOC 2,11              ″表示位置:  2行11列
90   PRD 3,I               ″数値表示:  変数Iを3桁表示
100  TIME 10
110 NEXT I
```

※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。

PRC,PRD,PRS参照

LP

ファイル　　点データ取り出し

■書 式
LP n
 0≦n≦3

■解 説
SPコマンドでメモリーエリアに保存した点データを取り出します。

SP,M参照

LSCNT

編集　　標準リスト行数設定

■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します。

```
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。
デフォルトは20です。

LIST参照

M

演算　　配列変数

■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。 M [n]<ent>でM配列の内容を表示します。 MPCINITでは初期化されません。

SP,LP参照

MIO

I/O　　MIO-816拡張

■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 816          ’MIO-816#6から#11を使う。
MIO 240          ’MIO-240,248を使う。出荷時はMIO-240モード
```

“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	

※()の中の数値はMIO-248の8点の入力ポート番号

MODE

パルス **パルス発生モード**

■書 式

MODE n

n:5または6

(PG -1 の時は無効)

■解 説

モードコマンドは、 MPG-303 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っ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 &H0A,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29.0kpps 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56.3kpps 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。

MODE	パルス幅	デューティー	最大パルスレート
5	20μSec	50%以上 *1	29.0pps
6	4.2μSec	50%以下	56.3pps

*1 ACCEL 25000以下の場合

PG参照

MON

デバッグ **停止行番号の参照**

■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。

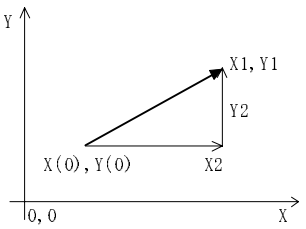
MOVE

パルス **XY絶対移動**

■書 式
MOVE X,Y
MOVE P(n)
X,Y:変数、 定数
n:点番号

■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)          ˝X(0) はX軸の現在座標
Y2=Y1-Y(0)          ˝Y(0) はY軸の現在座標
```



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。MODE 5、 MODE 6ともコマンド上の扱いは全く同等です。

RMOV,MOVZ参照

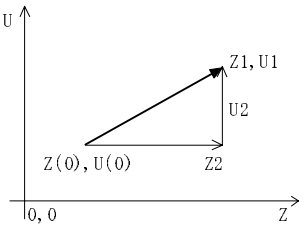
MOVZ

パルス **ZU絶対移動**

■書 式
MOVZ Z,U
MOVZ P(n)
Z,U:変数、 定数
n:点番号

■解 説
絶対座標移動。ZU軸ついてZ,U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 FEDZ でスピードを切り換える事ができます。MOVZ Z1,U1の場合のパルス発生量Z2,U2は

```
Z2=Z1-Z(0)          ˝Z(0) はZ軸の現在座標
U2=U1-U(0)          ˝U(0) はU軸の現在座標
```



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、 RMVZ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。MODE 5、 MODE 6ともコマンド上の扱いは全く同等です。

MOVE,RMVZ,RMOV参照

MPCINIT

メンテナンス　　MPC-816初期化

■書 式
MPCINIT

■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。
※ MPCINIT を実行すると点データや変数が消されます。 必要に応じて実行前にデータを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

ERASE参照

MTRX

パルス　　パレット分割量

■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)

PALET,PL,PLX,PLY参照

MTRX1

パルス　　パレット分割量

■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)

PALET1,PL1,PL1X,PL1Y参照

NEW

編集 **プログラムクリア**

■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。

FREE,NEWP参照

NEWP

編集 **点データクリア**

■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

NEW参照

NEXT

制御文 **繰り返しループ**

■書 式
FOR a1=a2 TO a3
NEXT a1
a1:変数
a2,a3:変数、 定数 a2<a3

■解 説
IF THEN,GOTOで飛びだし不可

FOR,DEC参照

O_IN

I/O　　出力状態の取得

■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参照

O_SW

I/O　　出力状態の取得

■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を返します。 メモリー I/OはSW(n)で読み出すことが出来ます。

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

O_IN参照

OFF

I/O　　ビット単位出力オフ

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

ON,OUT,O_SW参照

OFF_AND

I/O　　条件出力・論理積OFF

■書 式
OFF_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はオフです
>OFF_AND 1 0 0        ″OFF_ANDの結果が0の時には出力は保持
>PR O_SW(0)
 0                    ″オフのまま
>ON 0                 ″出力0をオン
>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
>PR O_SW(0)
 1                    ″オンのまま
>OFF_AND 1 1 0        ″OFF_AND結果が1になると出力0はオフ
>PR O_SW(0)
 0                    ″オフになった

OFF_AND SW(16),1,0    ″SW(16)がONならば出力0をOFFする
```

OFF_OR

I/O　　条件出力・論理和OFF

■書 式
OFF_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,A2のどちらかかが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

I/O　　ビット単位出力オン

■書 式
ON A1[,A2,A3]
A1,A2,A3：ポート番号
0≦A1,A2,A3≦255 I/O
-1≧A1,A2,A3≧-128 メモリー 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。出力はオープンコレクタの引き込み(負論理)なので、ON時はLOWレベル、OFF時はHIGHレベルですが、MIO-816#1～#5はSETIOコマンドで正論理に変更できます。

```
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
```

OFF,OUT,O_SW参照

ON_AND

I/O　　条件出力・論理積ON

■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をオンします
>PR O_SW(0)
1
>ON_AND 1 1 0          ″ON_AND結果が1なので出力はオンしたまま
>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
>PR O_SW(0)
1                      ″オフされません
>BL_AND 1 0 0          ″BL_ANDは結果が0になると出力はオフになる
>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 SET IO
20 WAIT SW(16)=1
30 ON_AND SW(17),1,-1           'SW(17)がONならば-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      'メモリー I/O同士→メモリI/O
55 ON_AND SW(-3),SW(-4),0       'メモリー I/O同士→物理I/O
60 PRINT O_SW(0)
70 WAIT SW(16)=0
80 GOTO 10
>RUN
 1
 1
 0
 0
```

この他にA1,A2には!SW(),HSW()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         'IF SW(16)=1 AND SW(17)=0 THEN ON 8 と同等
```

ON_OR

I/O　　条件出力・論理和ON

■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

A1,A2どちらかかが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

I/O　　パラレル出力

■書 式

OUT n,m

n:出力データ

0≦n≦255

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度にオン・オフします。

```
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
```

ボード	バンク	ビット
MPC-816	0	0～7
MIF-816	1	8～15
MIO-816#1	2	16～23
MIO-816#2	3	24～31
MIO-816#3	4	32～39
MIO-816#4	5	40～47
MIO-816#5	6	48～55

MemoryI/O -1	-1～-8
MemoryI/O -2	-9～-16
MemoryI/O -3	-17～-24
MemoryI/O -4	-25～-32
MemoryI/O -5	-33～-40
MemoryI/O -6	-41～-48
MemoryI/O -7	-49～-56
MemoryI/O -8	-57～-64
MemoryI/O -9	-65～-72
MemoryI/O -10	-73～-80
MemoryI/O -11	-81～-88
MemoryI/O -12	-89～-96
MemoryI/O -13	-97～-104
MemoryI/O -14	-105～-112
MemoryI/O -15	-113～-120
MemoryI/O -16	-121～-128

ON,OFF,O_SW参照

OUT_CNT

ソフトカウンタ　　比較出力付きカウンタ

■書 式
OUT_CNT

■解 説
MPC-816の入力を使った3byte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でカウント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でカウント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でカウント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。

```
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
```

※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1kpps程度です。

IO_CNT,OUT_CSET参照

OUT_CSET

ソフトカウンタ　　カウンタ設定

■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のビットパタン。(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

```
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：  in10はB接点検出となる
```

IO_CNT,OUT_CNT参照

OVRUN

パルス　　非常停止

■書 式

OVRUN n

n=&HLLMM (in24～31の入力パターン)

LL:論理パターン

MM:マスクパターン

※PG -1 の時

in24～31に対して入力停止機能を追加

■解 説

MIF-816 IN(3)のビット対応

bit	B7	B6	B5	B4	B3	B2	B1	B0
in	31	30	29	28	27	26	25	24

J=(IN(3) XOR LL) AND MM

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 &HFF          'in24から31のどれかがオンになれば停止
OVRUN &HFFFF        'in24から31のどれかがオフになれば停止
OVRUN &H81          'in24または31がオンで停止。他のポートは汎用入力
OVRUN &H8181        'in24または31がオフで停止。他のポートは汎用入力
```

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。 OVRUNが不要な場合はOVRUN 0と設定します。

■MPG-303を2枚使用した時のオーバーランコマンドについて

MPC-816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MPG-303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MPG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MPGの#1、タスク4でMPGの#2の制御をしていますがOVRUN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MPG#1は即停止しますが、 タスク4のMPG#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。 OVRUN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プ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プ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
 OVRUN &H000F           <-MPG#1は24～27がONで即停止する
*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
 OVRUN &H00F0           <-MPG#2は28～31がONで即停止する
*LOOP2
 RMOV 100000,100000
```

```
 TIME 50
 RMOV -100000,-100000
 TIME 50
 GOTO *LOOP2
                        □□□
10      OVRUN &H01      <-SW(24)をｵｰﾊﾞｰﾗﾝ入力設定
20      RMOV 100000,0
30      FOR I=0 TO 100
40       PRINT I
50       TIME 10
60      NEXT I
>RUN                    <-ﾌﾟﾛｸﾞﾗﾑを実行してSW(24)がONするとﾌﾟﾛｸﾞﾗﾑは
                          RMOVから抜けだし次のｽｯﾃﾌﾟを実行。
 0
 1
 2
 3
 4
>                       <-CTRL+Aでﾌﾟﾛｸﾞﾗﾑ停止
>PR SW(24)              <-ｵｰﾊﾞｰﾗﾝｾﾝｻｰが入った状態で再びRUNするとｴﾗｰ表示
 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-816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0をENDで終了させない）

P

パルス **点データ**

■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。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。

```
>SETP 1 111 222         'X(1),Y(1)セット
>STPZU 1 333 444        'Z(1),U(1)セット
>PLS                    '点データリスト表示
P(1): 111 222 333 444
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キーで止め。その他キーで継続
>PRINT P(1                  ″P(1)のX,Y,Z,U 表示
 111  222  333  444

>SETPOS 0 0                                  ″X,Y現在値設定
>STPZU 0 0 0                                 ″Z,U現在値設定
>T                                           ″ティーチモード
PG 1#1(X,Y,Z,U)  0  0  0  0 [XYZ,U] 500 500  ″現在値表示
>                                            ″Qキーで止め
```

※) Z版での4次元の座標表示は、 PRINT P(n),Z(n),U(n)

PALET

パルス　　パレット設定

■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MTRXと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。

```
PL(25) ->    PL(30)
P12  *+++++
     +++++
     +++++
     +++++
P10  *++++* P11
PL(1)  ->    PL(6)
```

```
PALET 10,11,12          ″ティーチングポイント
MTRX 6,5                ″分割数
  |
MOVE PL(n)              ″移動
```

パレット宣言によってPL(1)～PL(30)の点が使用できる様になります。 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。 (PALET1に対応するのはPL1(n),PL1X(n),PL1Y(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。 (Z及びUについてはP(i)のZU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。

```
10 PALET 10,11,10
20 MTRX 5,0
```

```
PL(1) PL(2) PL(3) PL(4) PL(5)
  *-----|-----|-----|-----*
P(10)                    P(11)
```

MTRX,PL,PLX,PLY参照

PALET1

パルス　　パレット設定

■書 式
PALET1 i,j,k
 i,j,k:ﾃｨｰﾁﾝｸﾞされた点の番号
  1≦i,j,k≦300

MTRX1,PL1,PL1X,PL1Y参照　　詳しくは PALET

PAUSE

タスク操作　　タスクの一時停

■書 式
PAUSE n[,m,l]
 n,m,l:タスク番号 1≦n,m,l≦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。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と扱い易くなります。

FORK,CONT,PAUSE参照

PG

パルス　　PGボード選択

■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 J5からパルス発生 (REV-3.53g以降)

■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。

TASK0	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。
TASK1～3	MPG#1 DSWを1に設定したものです。(出荷時)
TASK4～7	MPG#2 DSWを2に設定したものです。
TASK8～11	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
ACCEL 5000          "MPG#1に有効
FEED 0              "    〃
PG 2                "PGコマンドでMPGを切り替え
ACCEL 5000          "MPG#2に有効
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                    ″タスク1
MOVE 10000,10000        ″このタスクのパルス命令はMPG#1へ
WAIT BSY(0)=1
TIME 10
MOVE 0,0
WAIT BSY(0)=1
TIME 10
GOTO *PG1
*PG2                    ″タスク4
MOVE 10000,10000        ″このタスクのパルス命令はMPG#2へ
WAIT BSY(0)=1
TIME 10
MOVE 0,0
WAIT BSY(0)=1
TIME 10
GOTO *PG2
```

※MPG-303を使うにはPG宣言が必要です。
MPCINIT直後(デフォルト)は「PG -1」の状態になっています。この状態ではPG#1～#3のタスク引き当ても無視され、全てのタスクのパルスはMIF-816 J5から出力されます(PRG①)。
MPG-303を使う場合は(タスク0でPGを使わなくても)事前にPGコマンドを実行してください(PRG②)。
これはPG#2、 PG#3でも同様です③。
この操作はダイレクトコマンドでも有効ですが、 必ずプログラムへ記述してください。
PRG① (MPCINIT直後)

```
10  FORK 1,*PG          ″タスク1はPG#1から出力するはずが..
20  END
100 *PG
110 MODE 5
120 ACCEL 5000
130 SETPOS 0,0
140 MOVE 1000,1000      ″MIF J5からパルス出力となる
150 END
```

PRG②

```
5   PG 1                ″タスク0でPGコマンドを実行
10  FORK 1,*PG
20  END
100 *PG
110 MODE 5
120 ACCEL 5000
130 SETPOS 0,0
140 MOVE 1000,1000      ″PG#1からパルス発生
150 END
```

PRG③

```
5   PG 1                ″PG 1,2,3いずれかでOK
10  FORK 4,*PG          ″タスク4はPG#2にアクセス
20  END
100 *PG
110 MODE 5
120 ACCEL 5000
130 SETPOS 0,0
140 MOVE 1000,1000      ″PG#2からパルス発生
150 END
```

※1～11のタスクでPGコマンドを使うと、その効果はタスク0に反映されます。 1～11のタスクではPG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                 ″TASK0でPG2を宣言しても80行のPG1が有効になる
30 TIME 50
```

```
40 ACCEL 30000
50 FEED 0
60 MOVE 1000,0
70 *TASK1
80 PG 1
90 END
```

■n=-1のとき

PG -1とするとパルス出力ポートはMIF-816のJ5へ切り替わります。そのときSW(16)～(23)は原点センサー、SW(24)～(31)は停止入力ポート(OVRUN)となります。SHOM X,Y,s と SHMZ X,Y,s のsはPPS単位で設定(1～2000pps)します。STOP,JOG,PULSEコマンドはサポートしていません。MODEコマンドは無効です。座標は3byte長で、どのタスクからでもパルス発生が可能ですが、パルス発生中はマルチタスクが停止します。PG -1は<Ctrl>+<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)=1で停止。未使用SWは一般入力OK
'OVRUN &HFFFF                      ″SW(24)～(31)のどれかがOFFで停止
'OVRUN &HFF01                      ″SW(24)=0で停止。未使用SWは一般入力OK
GOSUB *HOME
*MAIN
GOSUB *CW
GOTO *MAIN
*CW
'CW
PRINT STR(-1)
TIME 5
FOR I0=1 TO 5
RMOV 1000 1000
RMVZ 1000 1000
GOSUB *OVRUNCHK
TIME 10
NEXT I0
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
'OVER RUN
PRINT STR(-1)
END
*HOME                              ″原点復帰
SHOM 1 4 400                       ″XCW YCW PPS
HOME &H0F 1000 1000                ″SW(16)～(19)=1,退避移動CW 1000
'                                  ″ 使わないSWは一般入力OK
'HOME COMP
PRINT STR(-1)
TIME 5
SHMZ 1 4 400                       ″UCW ZCW PPS
HOMZ &H0F 1000 1000                ″SW(17)～(23)=1,退避移動CW 1000
'                                  ″ 使わないSWは一般入力OK
'HOMZ COMP
PRINT STR(-1)
TIME 5
GOTO *__RTN
'===========
*__RTN
RETURN
```

※PG -1のパルス発生でOVRUN入力が入った場合は即停止(減速無し)となります。MPG-303のOVRUNはインタプリタが停止しますが、 PG -1では継続されます。

```
10      PG -1
20      ACCEL 5000
30      FEED 0
40      SETPOS 0,0
50      STPZU 0,0,0
60      OVRUN &HFF
70      MOVE 10000,10000
80      IF IN(3)<>0 THEN *OVRUN     ″もしもOVRUNなら
90      PRINT P(0)
100     END
110     *OVRUN
120     PRINT STR(-1)
RUN
 10000 10000 0 0                    ″正常停止=減速停止
RUN
 OVRUN                              ″OVRUN停止=即停止
```

PGS

パルス　　PGエラーの読み取り

■書 式

PGS(n)

0≦n≦3

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

■解 説

MPG-303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-303に発生したエラーを次のエラーまで保持しています。

PL

パルス　　パレット点データ

■書 式

PL(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
```

MTRX,PALET,PLX,PLY参照

PL1

パルス　　パレット点データ

■書 式
PL1(n)
 n:パレット上の点番号
  1≦n≦32767

MTRX1,PALET1,PL1X,PL1Y参照

PL1X

パルス　　パレット点のY成分

■書 式
PL1Y(n)
 n:パレット上の点番号
  1≦n≦32767

MTRX1,PALET1,PL1,PL1X参照

PL1Y

パルス　　パレット点のY成分

■書 式
PL1Y(n)
 n:パレット上の点番号
  1≦n≦32767

MTRX1,PALET1,PL1,PL1X参照

PLS

パルス　　点データリスト

■書 式
PLS n
 n=点番号

■解 説
点データP(n)の表示です。PLSで点1よりPLS nで点nより表示します。途中で一時休止しますが、中断の場合は'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
>
```

PLS_MIF

パルス　　パルスポート変更

■書 式
PLS_MIF

■解 説
ZPLS,YPLS,WPLS,VPLSの出力先は通常“out 0”ですが、このコマンドによってMIF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。

ZPLS,YPLS,WPLS,VPLS参照

PLX

パルス　　パレット点のX成分

■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
```

MTRX,PALET,PL,PLY参照

PLY

パルス　　パレット点のY成分

■書 式
PLY(n)
 n:パレット上の点番号
  1≦n≦32767

MTRX,PALET,PL,PLX参照

PR

デバッグ　　PRINTの短縮形

■書 式
PR A1[,A2,A3]
 A1,A2,A3:定数, 変数

■解 説
PRINTコマンドの省略形です。

PRINT,PRX参照

PRC

LCD　　キャラクタ表示

■書 式
PRC n[,m,l]
 n,m,l:キャラクタコード

■解 説
LCDにキャラクタを表示します。

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

LOC,PRD,PRS参照

PRD

LCD　　数値表示

■書 式
PRD n,m
 n:表示文字数
 m:変数

■解 説
LCDに変数の数値を表示します。

```
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)
```

LOC,PRC,PRS参照

PRINT

デバッグ　　数値データの表示

■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。

```
PRINT P(0)      ″点データを表示
PRINT SW(n)     ″ポートの値を表示
PRINT A1,B1     ″変数A1,B1の値を表示
PRINT IN(n)     ″パラレルデータの表示
```

PRX参照

PRINT#

RS-232C　　CH1からの通信出力

■書 式
PRINT# A1[,A2,A3]
A1,A2,A3:変数、定数

■解 説
RS-232C CH1からの数値出力です。デリミタは「スペース」(&H20)、ターミネーターは「CR･LF」です。

```
PRINT# 123   ″'123'[CR][LF] と出力します。
             ″ASCIIｺｰﾄﾞで言えば [&H31][&H32][&H33][&H0D][&H0A]
PRINT# A B   ″'(A)'[スペース]'(B)'[CR][LF] と出力します。
             ″ASCIIｺｰﾄﾞで言えば [(A)][&H20][(B)][&H0D][&H0A]
             ″(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PUT#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

LCD　　文字列表示

■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。

```
LOC 1,2
'acce|
PRS 5                ″1行2文字目から'acce|'(5文字)を表示
```

LOC,PRC,PRD参照

PRX

デバッグ　　ヘキサ表現

■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)
&H0F
PRX IN(5)
&HC0
```

PRINT参照

PULSE

パルス　　パルス出力

■書 式
PULSE n[,D1,D2]
 n:パルス発生数
  -32767≦n≦32767
 D1:ON時間（単位100μSEC）
  1≦D1≦32767
 D2:OFF時間（単位100μSEC）
  1≦D2≦32767

※PG -1の時はサポート無し

■解 説
固定デューティー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以下にすると誤差が大きい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1000CWパルス
PULSE -1000,100     'ON 10msec,OFF 10msecで1000CCWパルス
PULSE 2000,100,10   'ON 10msec,OFF 1msecで2000CW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

AXIS参照

PUT

RS-232C　　CH0からの1文字出力

■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に出力します。

```
PUT &H20              ″[スペース]を出力します。
PUT &H41              ″[A]と出力
PUT 65,66,67          ″[A][B][C]と出力
```

PUT#

RS-232C　　CH1からの1文字出力

■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に出力します。

```
PUT# &H20                 ″[スペース]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,&H0D    ″[D][E][CR]
40 I=I+1
50 AR(I)=GET#(0)          ″1文字入力
60 R1=RS(1)               ″CH1バッファのキャラクタ数
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J1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キャラクタずつ出力します。

```
CNFG# 4,0,2                ″ノンパリ,8ビット,9600bps
PUT# &H00B1,&H00B8,&H00BE  ″'ｱｸｾ'
PUT# &H00D9,&H000D         ″'ﾙ[CR]'
```

※PUT#は使用前にCNFG#でRS-232Cの初期化を行って下さい。

PUTS#

RS-232C　　CH1からの文字列出力

■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'[スペース]'456'[スペース]'789'と出力し[CR][LF]は出力しません。
PUTS# A B           ''(A)'[スペース]'(B)'と出力します。(A)(B)は変数Aの値の数字列です。
100 'AYAKO
110 PUTS# STR(100)  ''AYAKO'と出力されるのみ
```

QUIT

タスク操作　　タスクの停止

■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(アクセス中)のタスクには直接QUITを実施しないで下さい。必ずSTOPやBSY(n)を併用して下さい。

FORK,CONT,PAUSE参照

REG

MPG-301　　X3202レジスタ読込み

■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:MPG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読込
    -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込みます。 REG()は、1または2byteのレジスタ読込みです。 3byte符号付きは読込みはREG3()です。

```
WAIT REG(-1)=&H20        "MPG-301 #1 動作完了待ち
```

詳細：「MPG-301 製品別マニュアル」

CMND,REG3(),ST REG参照

REG3

MPG-301　　X3202レジスタ読込み

■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込みます。 REG3()は3byte符号付き読込みです。

```
C=REG3(&H0121)        ″カウンタA読込み
```

詳細：「MPG-301 製品別マニュアル」

CMND,REG(),ST REG参照

REM

編集　　コメント

■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。REMと入力してもシングルクォートに置き変わります。コメント文はSTR()関数の文字列定数として使用できます。

```
100 ' COMMENT
110 PRINT STR(-1)      ″「COMMENT」をPC画面に表示します
```

RENUM

編集　　文番号の編集

■書 式
RENUM [n,o,s]
 n:新文番号 0≦n≦32766
 o:旧文番号 0≦o≦32766
 s:ステップ量 1≦s≦1000

■解 説

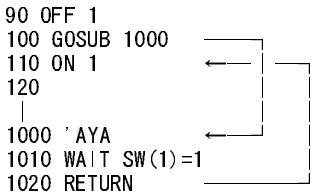
```
RENUM                 ″最初から番号10番おきにふり直し
RENUM 100             ″ふり直し開始番号が100となる
RENUM 100,50          ″50番以降を開始番号100で10番おきにふり直し
RENUM 100,50,5        ″ステップ量を5にする
```

RETURN

制御文 **サブルーチンリターン**

■書 式
RETURN

■解 説
サブールチンコールはRETURN文により元の実行ルーチンに戻ります。



この様に、RETURNはGOSUB次にジャンプします。GOSUBのネスト(多重コール)は、FOR文と併せて24レベルまでです。

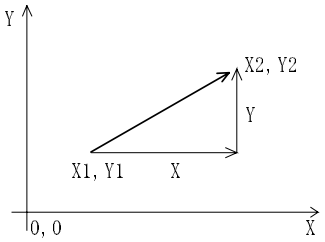
GOSUB参照

RMOV

パルス **XY相対座標移動**

■書 式
RMOV X,Y
X,Y:定数、 変数

■解 説
XY軸に対してX,Yパルス出力します。 直線補間、 相対座標移動です。 モード5,6ともコマンド上の扱いは全く同等です。最大スピードはACCEL、スピードの変更はFEEDコマンドによって行います。移動開始位置をX1,Y1とするとパルス発生後の位置X2,Y2は次の通りです。



X2=X1+X
Y2=Y1+Y

途中で停止する場合はＳＴＯＰコマンドを使用します。

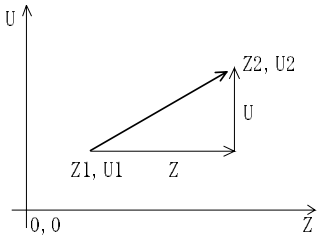
MOVE,RMVZ,MOVZ,D45参照

RMVZ

パルス　　ZU相対座標移動

■書 式
RMVZ Z,U
 Z,U:定数、 変数

■解 説
ZU軸に対してZ,Uパルス出力します。 直線補間、 相対座標移動です。 モード5,6ともコマンド上の扱いは全く同等です。最大スピードはACCEL、スピードの変更はFEDZコマンドによって行います。移動開始位置をZ1,U1とするとパルス発生後の位置Z2,U2は次の通りです。



Z2=Z1+Z
U2=U1+U

途中で停止する場合はＳＴＯＰコマンドを使用します。

RMOV,MOVE,RMVZ,D45参照

RS

RS-232C　　受信キャラクタ数

■書 式
RS(n)
 n:RS-232C番号 0または1
  0:プログラムポート
  1:CH1(ユーザー)

■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。

```
PRINT RS(1)
12                    ″バッファに12個有る
```

CH1の入力バッファは64バイトです。

RSV

タスク操作　　セマフォ

■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ー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          ″セマフォ取得
MOVE 0,0
OFF -1                  ″セマフォ開放
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MOVE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

制御文 **プログラム実行**

■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

プログラムを実行します。 プログラム編集直後はFROMに書き込んでから実行します。 実行終了番号は指定できません。 途中で実行を中止する場合は、BRK,END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リセットボタンを押すか、キーボードで<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です。

```
10        I=0
20       *LOOP
30        I=I+1
40       PRINT I
50       TIME 50
60       GOTO *LOOP
RUN                          ″(このRUNは上書されて実際には見えない)
Programming the FLASH ROM *+++++++++    ″FROM書込み
 1
 2
 3
 4
                             ″<Ctr|>+<A>で停止
      TASK 0 # 50            ″停止位置のタスク番号と文番号
>
TASK0 50         TIME 50     ″<Ctr|>+<M>でリスト表示
```

BRK,END,FWRITE,TON,TOFF参照

SET

パルス　　JOG量設定

■書 式
SET n,i,j
 n:バンク
  0≦n≦3
 i:XYZのJOG量
  -32767≦i≦32767
 j:UのJOG量
  -32767≦j≦32767

■解 説
ティーチモードのJOG量を設定します。 TEACH(T)コマンド実行時には 0～3 までのキーにより移動量を選択できますが、その値はこのSETコマンドで設定します。

```
>SET 0 10 10
>SET 1 20 20
```

TEACH参照

SETIO

I/O　　出力初期化

■書 式
SETIO

■解 説
出力の初期化に使用します。

```
>SETIO          ″全部の出力を一括でオフします。
```

※メモリー I/Oもオフになります。

SETP

パルス　　点データの設定

■書 式

SETP n,x,y

n:点番号(1～300)

x,y：変数、定数

-8388607≦x,y≦8388607

■解 説

P(n)に座標x,yを入力します。nを0とすると現在位置の意味となります。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      ″P(80)のXを100,Yを200
STPZU 80,300,400     ″P(80)のZを300,Uを400
SETP 81,X(0),Y(0)    ″P(81)のX,Yを現在点
STPZU 81,Z(0),U(0)   ″P(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    ″P(100)のYだけ0
```

STPZU,SETPOS参照

SETPOS

パルス　　現在位置の設定

■書 式

SETPOS x,y

x,y:変数、定数

-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。

SETP,STPZU参照

SETVAR

演算 **変数の初期化**

■書 式

SETVAR V1,V2[,n]

V1,V2:変数

n:変数、 定数(省略時0)

■解 説

MPC-816の変数は下表の様に配列されています。V1には低い方の変数名、V2には高い方の変数名、nには初期化する値を設定するとV1とV2の間の変数がnになります。 nを省略すると0で初期化されます。

変数の配列

```
低
 A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A
 B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 B
 C0 C1    |         |
          |         |      X8 X9 X
 Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7 Y8 Y9 Y
 Z0 Z1 Z2 Z3 Z4 Z5 Z6 Z7 Z8 Z9 Z
                                高
```

```
SETVAR A0 Z 999        ″A0からZまで、つまり全部を999にする
SETVAR B0 B 0          ″B行の変数を0にする
SETVAR C2 C9           ″C2,C3,C4,C5,C6,C7,C8,C9を0する
A=123
SETVAR D0 D A          ″D行の変数を123にする
```

VLIST参照

SFDL

パルス **Z軸スピード設定**

■書 式

SFDL n

n:加速パルス数

1≦n≦加速パルス数(ACCELで設定)

■解 説

FEDZの0～15は最高スピードを等分割して得ますが、FEDZ 15より更に遅いスピードを必要とする場合にはSFDLで加速パルス数を強制的に変更することができます。 著しく遅いスピードを得ることができます。

```
100 SFDL 1
110 FEDZ 15
120 RMVZ 1000
```

上記は加速距離1パルスのため全く加速しません。 加速パルス数についてはACCELを参照下さい。

SFTL

演算 AR(n)の左シフト

■書 式
SFTL

■解 説
AR(n)のデータを次の様にローテーションします。
→AR(2)→AR(1)→AR(0)→AR(31)→
AR(n)をシフトレジスタとして使用することができます。

AR(),SFTR参照

SFTR

演算 AR(n)の右シフト

■書 式
SFTR

■解 説
AR(n)のデータを次の様にローテーションします。
→AR(31)→AR(0)→AR(1)→AR(2)→
AR(n)をシフトレジスタとして使用することができます。

AR(),SFTL参照

SHMZ

パルス 原点復帰設定

■書 式
SHMZ u,z,s
u:U軸原点復帰方向
1(CW),2(CCW),0(無し)
z:Z軸原点復帰方向
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単位で設定1～2000pps

■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
```

HOMZ参照

SHOM

パルス **原点復帰設定**

■書 式
SHOM x,y,s
x:X軸原点復帰方向
1(CW),2(CCW),0(無し)
y:Y軸原点復帰方向
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単位で設定1～2000pps

■解 説
原点復帰HOME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
```

sと原点復帰時のパルスレートの関係は次の通りです。
MODE 5 及び MODE 6
F=8×10e6/(413+(52×s+351)×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。

HOME参照

SHRD

パルス **原点センサー比較回数**

■書 式
SHRD n
2≦n≦10 (初期値4)

■解 説
HOME コマンドはパルスを出力する毎に原点入力をモニターし指定されたセンサーパタンと比較します。この比較回数を定めるのがSHRDです。サーボモータードライバーなどの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な場合があります。その時はSHRD 6～SHRD 8を指定し検出を確実なものとします。nの値と原点復帰スピードの関係はSHOM参照

HOME参照

SKIP#

RS-232C　　文字コード検索

■書 式

SKIP# n

n:アスキーコード

0≦n≦&H7F

■解 説

RS-232C CH1の受信バッファ先頭からnまでを読み捨てます。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 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(&H3Dは'='です)。この後、GET#を実行すると'123'という数値を得ることができます。

```
受信バッファ先頭
  ↓
 A=123
  ↑ SKIP# &H3D は'='まで読み捨てる
```

SKPSP#

RS-232C　　スペースコードの検索

■書 式

SKPSP#

■解 説

RS-232C CH1の受信バッファ先頭からスペースまでを読み捨てます。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。

SLOW

メンテナンス　　送出スピード設定

■書 式

SLOW n

n:文字出力タイマ

（1msec単位） 0≦n≦15

■解 説

MPC-816K以降無効(Rev-3.53sで確認)

SP

ファイル　　点データ保存

■書 式

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)と共通になってい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。保存先はSRAMです(フラッシュ ROMではありません)。

```
10 FOR I=1 TO 10
20 J=I*100
30 K=I*1000
40 SETP I,J,K          ″ポイントデータ作成
50 NEXT I
>RUN
>PLS                   ″確認表示
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                  ″ポイントデータ保存
>NEWP                  ″メインメモリーのデータ消去
>PLS                   ″確認表示
P(1):  0 0 0 0         ″┐
P(2):  0 0 0 0         ″│ 消えている
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                  ″ポイントデータ読込
>PLS                   ″確認表示
P(1):  100 1000 0 0    ″┐
P(2):  200 2000 0 0    ″│ もとに戻った
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
>
```

LP,M参照

SQR

演算 **平方根の算出**

■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

演算 **平方根**

■書 式
SQRT n m v
n,m:変数,定数 >0
v:結果が入る変数

■解 説
n,mの平方根を求めます。(v=root(n*n+m*m))

```
>SQRT 3000 4000 V0
>PR V0
5000
```

ST_REG

MPG-301 **X3202レジスタ書込**

■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。

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

参照:「MPG-301 製品別マニュアル」

CMND,REG(),REG3()参照

STOP

パルス　　MPG-303パルス停止

■書 式

STOP n

n:停止モード

1=減速停止

2=急停止

3=I/O指定

4=MPG停止

5=MPG停止解除

※PG -1でのパルス発生はSTOPできません

■解 説

■n=1,2について

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について

STOP 3,P,O

P:ポート番号

-128≦P≦512 (#DEFS定義の文字列は不可,変数は可)

O:I/O状態

0=OFF,1=ON

STOP 3,P,O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        "SW(16)がONで減速停止に設定
110 MOVE 20000,20000   "STOP設定直後のコマンドに有効
120 MOVE 0,0           "この行には無効
```

STOP 3 はJOGには利きません。 JOGは実行後直ちに次の行に移行するのでSTOP 3は無効です。

```
10 PG 1
20 MODE 5
30 ACCEL 5000
40 STOP 3,0,1          " 「SW(0)がONで停止」という設定
50 JOG 50,1            "PRGはこのJOGを実行してすぐ下へ
60 WAIT SW(1)=1        "PRGはここで入力待ちとなり
70 STOP 2              "SW(1)がONになればこのSTOPで停止
```

■STOP 4,5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終了後に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QIOT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              "このタスクのパルス機能停止
20 PG 1                "PGコマンドは大丈夫
30 MODE 6              "STOP 4状態ではハングアップする
```

■STOPコマンド使用上の注意

STOP,JOG,X(0),Y(0),BSY(0)などのパルス発生コマンドや現在点取得関数はMPG-303にアクセスします。 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がある=OK
  120 STOP 1
  130 Y=Y(0)           "STOPの後にすぐ現在点を取得=NG
      "↑
      "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)
```

※STOPコマンド後の出力されたパルス数と座標値は整合されています。
※STOPはHOME,PULSEコマンドにも有効です。
※パルス発生中のタスクをPAUSE,QUITする場合はSTOPとBSY()で停止確認。
※HOMEに対してもSTOPは有効ですがSTOP 1でも2でも急停止。 また、 BSYの戻り値は常に1です。

BSY()参照

STPZU

パルス　　点データ設定

■書 式
STPZU n,z,u
 n:点番号
  0≦n≦300 (0は現在点)
 z:Z座標の値(変数,定数)
 u:U座標の値(変数,定数)

■解 説
P(n)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。 nを0とすると現在位置設定となります。

SETP,SETPOS参照

STR

文字列　　文字列の呼出

■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、n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'       ″この文字列を出力したい
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
```

※コメントは12文字以下、日本語不可
※存在しない文番号や-2以下の数字を指定するとVOIDになります。

```
10      'ABC
20      PRINT STR(5)
30      PRINT STR(-2)
RUN
  VOID
  VOID
```

SW

I/O　　ビット単位入力

■書 式
SW(n)
 n:ポート番号
  0≦n≦255 I/O
  -128≦n≦-1 メモリー I/O

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

```
WAIT SW(5)=1            ″ポート5がONになるまで待つ
```

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てその値を返します。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。

```
実行スピード測定サンプル
    FOR I=1 TO 1000
       A=SW(0)
    NEXT I
＜実行結果＞
    816K:100mSec =0.1mSec/1回
    816X:5006mSec=5mSec/1回
```

※MBK-816の入力で

```
IF SW(50000)=1 THEN *ABC
```

と書くと!! Too Long} エラーになります。回避策は

```
A0=50000
IF SW(A0)=1 THEN *ABC      ″50000を変数にして文を短くする
```

IN,HSW,!SW参照

!SW

I/O　　SWの論理反転

■書 式
!SW(n)
 n:ポート番号
  0≦n≦255 I/O
  -128≦n≦-1 メモリー I/O

SW参照

SYS

タイマー　　システムクロック読込

■書 式
SYS(n)
 n=0 1m秒カウント値
 n<>0 カウント値をnで除算して返す
  (SYS(1000)とすると秒単位)

■解 説
パワーオン後から1m秒単位でカウントしている内部タイマーを読み取ります。
内部タイマーは0～&H1FFFFFFFで、最大約149時間(6.2日)までカウント可能です。ソフトリセットはできません。

```
>PR SYS(0)
 10679        ″パワーオンから約10秒経過
>PR SYS(2)
 8015         ″カウンタ値/2 (少数点以下切り捨て)
```

これはソフトタイマーです。マルチタスク停止中はタイマーのカウントも停止します。例えば、PG -1でハルス発生を行っている間は進みません。

MPC-816の変数は符号付き3Byte長なので&H7FFFFFを超える値をロードすると負の値となります。
例えば、SYS(0)(mSec単位)で変数にロードできるのは最大 &H7FFFFF = 8388607ms ≒ 139m ≒ 2.3時間まで、SYS(1000)(Sec単位)とすれば内部タイマー最大値の536870秒まで扱えます。

```
>T0=SYS(0) ″ power on 後約2時間半経過
>PR T0
 -7869273   ″ &H87ECA7 = 8907943 オーバーフロー
>T0=SYS(1000)
>PR T0
 8908        ″ 1/1000すればオーバーフローしない
```

※内部タイマー値がオーバーフローしたときの値は不定。
※内部タイマーには数%の誤差があります。 また、原発のセラロックにも周波数精度±0.5%、 温度安定性±0.2%の誤差がありますので、 必要に応じてボード毎に補正してください。
※サポートはRev-3.53以降

T

パルス　　TEACHの短縮形

■書 式
T

TEACH参照

TAIL

編集　　プログラム最終番号表示

■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

タスク操作　　タスクの状態監視

■書 式
TASK(n)
 n:タスク番号
  -1≦n≦11
   -1:自分のタスク番号取得
   0～11:タスクの状態

■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返します。
　　戻り値
　　0:実行中
　　4:一時停止中（PAUSE状態）
　　7:未使用（QUIT状態）
例えばコマンドで実行すると次の様になります。

```
>PRINT TASK(-1)          ″自分のタスク番号は?
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、1は未使用
>
```

■リセット状態の判別
n=15でMPCの再起動状態を知ることができます。
TASK(15)
　　戻り値
　　0:電源オンによる起動
　　1:リセットSWによる起動
　　2:バッテリエラー

TEACH

パルス　　ティーチモード

■書 式
TEACH（省略形 T）

■解 説
MPC-816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はTまたはTEACHと入力します。この時、現在位置、インチング量がそれぞれ表示されます。インチング量はSETコマンドによって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。

キー	動作
[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方向へインチング量だけ移動します。
[x]	-X方向へインチング量だけ移動します。
[Y]	+Y方向へインチング量だけ移動します。
[y]	-Y方向へインチング量だけ移動します。
[U]	+U方向へインチング量だけ移動します。
[u]	-U方向へインチング量だけ移動します。
[Z]	+Z方向へインチング量だけ移動します。
[z]	-Z方向へ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'が押されると点番号の入力モードとなります。改行があり'P'表示の後に番号を入力すると現在位置を憶えます。
[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)

※インチングとはキーを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,SETP,STPZU等を使いプログラムします。
※MPCINIT直後は「PG -1」状態(MIF-816 J5からパルス発生)になっています。MPG-303を使う場合は事前に「PG n」コマンドを実行して下さい。 PGはプログラムにも記述して下さい。　PG参照

■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番号                              JOG量。0,1,2,3キー
"TABで切替          座標値           で変わります。
"  ┌──┐          ┌──────┐  ┌──────┐
 PG 1#1(X,Y,Z,U) 0 0 0 0 [XYZ,U] 400 400
 "          ↓   Xキーをイッパツ
 PG 1#1(X,Y,Z,U) 400 0 0 0 [XYZ,U] 400 400
 "          ↓   Xキーをもうイッパツ
 PG 1#1(X,Y,Z,U) 800 0 0 0 [XYZ,U] 400 400
 "          ↓   Xキーをさらにイッパツ
 PG 1#1(X,Y,Z,U) 1200 0 0 0 [XYZ,U] 400 400
 "          ↓   Zキーをイッパツ
 PG 1#1(X,Y,Z,U) 1200 0 400 0 [XYZ,U] 400 400
 "          ↓   Zキーをもうイッパツ
 PG 1#1(X,Y,Z,U) 1200 0 800 0 [XYZ,U] 400 400
 "          ↓   Zキーをさらにイッパツ
 PG 1#1(X,Y,Z,U) 1200 0 1200 0 [XYZ,U] 400 400
 "          ↓   Zキーをついでにイッパツ
 PG 1#1(X,Y,Z,U) 1200 0 1600 0 [XYZ,U] 400 400
 "          ↓   Zキーをおまけにイッパツ
```

```
PG 1#1(X,Y,Z,U) 1200 0 2000 0 [XYZ,U] 400 400
"          ↓
P1  "確定位置で'P'キーを押し番号を入力=ここがP(1)
"          ↓
PG 1#1(X,Y,Z,U) 1200 0 2000 0 [XYZ,U] 400 400
"          ↓   Xキーを5ハツ
PG 1#1(X,Y,Z,U) 3200 0 2000 0 [XYZ,U] 400 400
"          ↓   Zキーを4ハツ
PG 1#1(X,Y,Z,U) 3200 0 3600 0 [XYZ,U] 400 400
"          ↓
P2  "確定位置で'P'キーを押し番号を入力=ここがP(2)
"          ↓
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キー
PLS     "点データの一覧を表示
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)に移動
35 MOVZ P(1)
40 PRINT P(0) "現在の座標値を表示
50 TIME 10
60 MOVE P(2)  "ティーチングしたP(2)に移動
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

制御文 **条件分岐**

■書 式

IF～条件式～THEN n

IF～条件式～GOSUB n

n:文番号 1≦n≦32766 またはラベル

IF参照

TIME

タイマー　　ディレイタイマー

■書 式
TIME n
 n:タイマー値
  0≦n≦8388607

■解 説
時間待ちのタイマーです。 10msec単位です。

```
TIME 100 (100×10msec=1000msec=1秒)
```

引き数に X(n)、 M(n)などの配列は使えません。 定数または変数にして下さい。

```
TO=X(1)
 TIME TO
```

TMOUT

タイマー　　タイムアウト設定

■書 式
TMOUT n
 n:タイマー値
  0≦n≦8388607

■解 説
WS1(),WS0()のタイムアウト時間を設定します。10msec単位です。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WS1(),WS0()関数に適応されます。
※タイムアウト付き関数実行中に別のタスクでTMOUT値を変更しても待ち時間は変わらない。

WS1,WS0,HWS1,HWS0参照

TOFF

デバッグ　　トレースモード解除

■書 式
TOFF

■解 説
タスク0でのみ有効

TON参照

TON

デバッグ　　トレースモード設定

■書 式
TON

■解 説
メインタスク(タスク0)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。

操作キー	動作
<Space>	1ステップ実行
<Ctrl>＋<C>	中止
<Q>	コマンドモードへ復帰
その他のキー	継続

<Q>キー操作後はCNTで再実行、<Ctrl>+<C>で中止します。TONモードの解除はハードリセットかTOFFを実行します。

TOFF参照

TST#

RS-232C　　キャラクタの分類

■書 式
TST#(n)
n:ダミー(0)を設定

■解 説
RS-232C CH1バッファの先頭文字がどの様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。

戻り値
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'という文字列があるとき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

パルス　　U軸座標取り出し

■書 式
U(n)
 n:点番号
  0≦n≦300

■解 説
U座標値を返します。 n=0の時は現在位置となります。

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

上は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、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
```

※MPG-303が無い状態でU(0)を実行すると !! PG Comunication error

URANG

パルス　　JOG領域設定

■書 式
URANG max,min
 max:上限
 min:下限
  -8388607≦min<max≦8388607

■解 説
初期値は不定です。 プログラムで初期化して下さい。

XRANG,YRANG,ZRANG参照

V_SWAP

メンテナンス　　版切り替え

■書 式
V_SWAP

■解 説
P/Z版を切り替えます。

```
TNYFSC(R) Rev-3.53s [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
*
```

※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。

VER

メンテナンス　　バージョン表示

■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。VER 0とすればソフトの履歴を表示します。

VLIST

デバッグ　　変数の一覧表示

■書 式
VLIST

■解 説
A0～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>キーで中断、その他のキーで継続
>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
>
```

SETVAR参照

VPLS

パルス **拡張パルス出力**

■書 式

VPLS p r c

p:出力ポート(out0かMIF-816のJ5)

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

2.17μ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) 出力パルス数を記録する変数

PLS,MIF,WPLS参照

WAIT

制御文 **条件待ち**

■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 変数Aが1になるまで待つ

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IF文で使用されるものと同一です。

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が大きいか等しい

WAIT A<=B AよりBが大きいか等しい

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

パルス　　拡張パルス出力

■書 式

WPLS p r c

p:出力ポート(out0かMIF-816のJ5)

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

2.17μ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) 出力パルス数を記録する変数

■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
```

前記の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2.17μ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。停止方法と停止待ち方法はYPLSと同様です。パルス出力は初期状態でout0(MPC-816の出力)からですがPLS_MIFでMIF-816 J5コネクタに変更できます。

```
PLS_MIF                 "出力をMIFに変更
FORK 1,*TASK1
'===========
*LOOP0
'direction
A0=1                    "方向 X-CW
'counter
C0=0                    "カウンタクリア
'
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,VPLS参照

WS0

I/O　　タイムアウト付入力

■書 式

WS0(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

これら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ます。
WAITの条件が成立すれば0、タイムアウトならば1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     ″メモリI/O -1 をオフ
>PR SW(-1)  ″確認
 0
>RUN        ″PRG実行
 TMOUT      ″5秒内に条件不成立で1を返し*TMOUTを実行
>ON -1      ″メモリI/O -1 をオン
>PR SW(-1)  ″確認
 1
>RUN        ″PRG実行
 OK         ″条件成立しているので0を返しIF文の次行へ。
```

TMOUT,WS1参照

WS1

I/O　　タイムアウト付入力

■書 式
WS1(n)
 n:ポート番号
  0≦n≦255 I/O
  -128≦n≦-1 メモリー I/O

■解 説
  WS0(n) WAIT SW(n)=0
  WS1(n) WAIT SW(n)=1
タイムアウト時間はTMOUTで設定。
  戻り値
   0:条件成立
   1:タイムアウト

TMOUT,WS0参照

X

パルス　　X軸座標取り出し

■書 式
X(n)
 n:点番号
  0≦n≦300

■解 説
X座標値を返します。 n=0の時は現在位置となります。

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

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)
```

P(10)を現在位置のX,Y値に設定します。 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
```

※MPG-303が無い状態でX(0)を実行すると !! PG Comunication error

XRANG

パルス　　JOG領域設定

■書 式
XRANG max,min
 max:上限
 min:下限
  -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の位置で停止します。 この計算方法は次の通りです。 整数で演算します。

```
n'=(n/256)×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。初期値は不定です。プログラムで初期化して下さい。

YRANG,URANG,ZRANG参照

Y

パルス　　Y座標取り出し

■書 式
Y(n)
 n:点番号
  0≦n≦300

■解 説
Y座標値を返します。 n=0の時は現在位置となります。

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

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)
```

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

```
Y(1)=100
Y(2)=Y(3)+A1
```

※MPG-303が無い状態でY(0)を実行すると !! PG Comunication error

YPLS

パルス　　拡張パルス出力

■書 式

YPLS p,r,c

p:出力ポート(out0かMIF-816のJ5)

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:パルスレート

10pps～3000pps程度

c:パルスカウンター(省略可) 出力パルス数を記録する変数

■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。ZPLSはYPLSと同等のコマンドでYPLS,ZPLS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256になるのを待ちます。パルスはデューティー50%の正方波形です。パルス出力は初期状態でout0(MPC-816の出力)からですがPLS_MIFでMIF-816 J5コネクタに変更できます。

```
10 PLS_MIF            'MIF J5からパルス出力
20  A1=1              'X軸CW駆動
30  A2=1000           'パルスレート=約1Kpps
40  A3=0              'カウンタクリア
50 YPLS A1,A2,A3      'パルス出力
60 WAIT SW(0)=1
70  A1=0              'パルス停止
80 WAIT A1=256        '停止待ち
90 PRINT A3           'カウンタ表示
```

PLS_MIF,ZPLS参照

YRANG

パルス　　JOG領域設定

■書 式

YRANG max,min

max:上限

min:下限

-8388607≦min<max≦8388607

■解 説

初期値は不定です。プログラムで初期化して下さい。

XRANG,URANG,ZRANG参照

Z

パルス　　Z軸座標取り出し

■書 式
Z(n)
 n:点番号
  0≦n≦300

■解 説
Z座標値を返します。 n=0の時は現在位置となります。

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

P(3)のZの値が100であることを表示しています。

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

P(10)を現在位置のZ,U値に設定します。 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
```

注)MPG-303が無い状態でZ(0)を実行すると !! PG Comunication error

ZPLS

パルス　　拡張パルス出力

■書 式
ZPLS p,r,c
 p:出力ポート(out0かMIF-816のJ5)
  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:パルスレート
  10pps～3000pps程度
 c:パルスカウンター(省略可) 出力パルス数を記録する変数

PLS,MIF,YPLS参照

ZRANG

パルス　　JOG領域設定

■書 式
ZRANG max,min
 max:上限
 min:下限
  -8388607≦min<max≦8388607

■解 説
初期値は不定です。 プログラムで初期化して下さい。

XRANG,YRANG,URANG参照