第５章　コマンド・リファレンス

５．１　変数と定数

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

１）定数

定数は、１０進と１６進が用意されており、それぞれ次の様に表現できます。

```
1234     &H123
```

実際には、次の様に使用されます。

```
A=1234・・・・・・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。

２）変数

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

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

以上２８６個の変数が使用できます。

３）配列

Ｐ版では、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。

４）関数

関数は、入力系のコマンドに利用されています。例えば、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。
```

５）演算

演算・代入は、式により実施されます。

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

演算: D0=A+B
      D0=123+B
```

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

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

A=SW(0)^B,A=SW(0)|Bということも出来ます。

６）制御文

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

７）実行中のエラー

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

参照コマンド ERR_ON

５．２　コマンド・リファレンス

【＊ラベル】　　機能：制御文　　サポート：Ｐ・Ｚ

■書　式　　*ﾗﾍﾞﾙ

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

文番号　＊ＬＡＢＥＬ　（１０文字以内）

通常のリスト表示の場合にはラベルのみ表示しますが、ＦＴＭ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。
ＦＴＭ(ＣＲＴ)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
```

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

【Ａ＿ＳＴＡＲＴ】　機能：制御文　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　　A_START 0
A_START n
　　n：時間(sec)
　　9≦n≦255

■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はターミナルケーブルが接続されているとターミナルモードに、ケーブルを抜くか、接続せずにパワーオンすると自動的にプログラムが実行されます。ケーブルが接続されていてもＡ＿ＳＴＡＲＴコマンドにより一定時間経過後にプログラムを実行することも可能です。
◇Ａ＿ＳＴＡＲＴで自動実行を行うときはプログラムにＡ＿ＳＴＡＲＴコマンドを追加した後１度プログラムを実行して下さい。

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

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

【ＡＣＣＥＬ】　機能：パルス　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　ACCEL n[,m,i]

n：最大スピード(pps)
m：加速距離(pls)
ｉ：オフセットスピード(pps)

Ｐ版　200≦n≦30000(MODE 5)
2000≦n≦50000(MODE 6)
m≦5000（省略するとm=n/10）
0<1/2n

Ｚ版　200≦n≦32000
200≦m≦2040
0≦i≦n

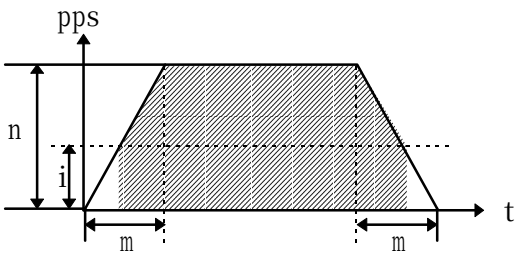
※ｎ及びmの下２桁は無効です。精度は、１００毎です。また、ＦＥＥＤのスピード値も正確なものではありません。

■解　説　ｎ，ｍでＦＥＥＤ　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４００パルスですから、１－２相駆動のステップモータにて約１回転で４０００ppsに到達することになります。これを加速度に換算すると次の様になります。

加速度＝$\frac{n^2}{2\times m}$ pps/(sec)^2　この場合では、$\frac{(4000)^2}{2\times 400}=20k/(sec)^2$　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、$\frac{2\times 400}{4000}=0.2\sec$　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
```

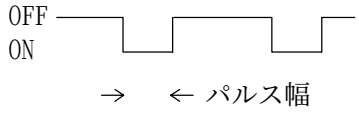
```
80  MOVE 0,0
90  GOTO *LOOP
```

◇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 ＲＥＶ　３．２（Ｐ版）				
	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	20 μ sec		4 μ sec	

MODE 6ではn＜2000の設定はできません。

ＭＩＦ－８１６からパルス発生 ＲＥＶ　２．２０ａ（Ｚ版）								
	MODE1		MODE2		MODE3		MODE4	
n	MOVE	JOG	MOVE	JOG	MOVE	JOG	MOVE	JOG
32000	31.6	12.5	33.1	17.1	62	20.5	31.2	11.6
30000	29.5	11.6	31.1	15.5	57.6	19	29.3	10.8
25000	24.8	9.4	25	11.4	47	15.5	24.6	8.9
20000	19.5	7.2	20.1	8.5	37.8	12.5	19.4	6.9
15000	15.1	5.4	15.3	6.1	28.1	9.3	14.5	5.1
10000	9.9	1.5	10.1	3.8	18.6	6.2	9.9	3.4
8000	8	2.8	8.1	2.9	14.7	4.9	7.8	2.7
4000	4	1.4	4	1.4	7.3	1.4	4	1.3
2000	2	0.7	2	0.7	3.7	1.2	2	0.7
ﾊﾟﾙｽ幅	8 μ sec		15 μ sec		最小 8 μ sec		10 μ sec	



◇極低速のパルス出力方法

Ｐ版の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。

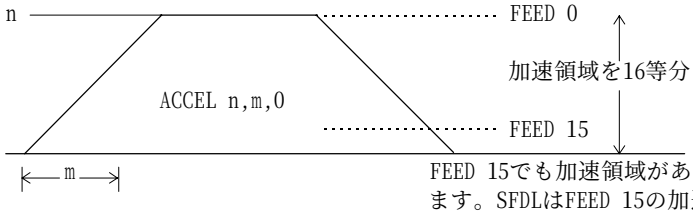
ＲＭＯＶの後にタイマーを用いればどんなに遅いパルスも出力できます。

```
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
```



FEED 15でも加速領域があります。SFDLはFEED 15の加速パルスを強制的に変更します。

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。

【ＡＤ】　　機能：ＭＩＦ－８１６ＡＤ　　種別：関数　　サポート：Ｐ

■書式　AD(n)

n：入力ｉｎ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で１ｍＶ単位です。

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

【ＡＲ】　　機能：演算　　種別：配列　　サポート：Ｐ・Ｚ

■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(0)=IN(0)
AR(1)=5
PRINT AR(0)
```

◇ＡＲ(ｎ)はＳＦＴＲ，ＳＦＴＬコマンドでデータのシフトができます。

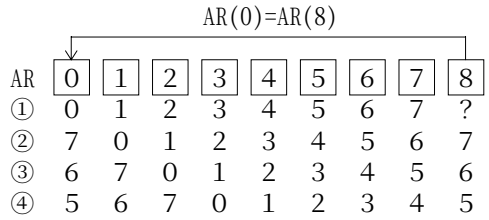
◇配列変数ＡＲ(ｎ)とＳＦＴＲの使用例

```
10      FOR I=0 TO 7
20        AR(I)=I     <-配列変数 AR(0)～AR(7)にﾃﾞｰﾀを入れます。
30      NEXT I
40      '
50      *LOOP
60      PRINT AR(0),AR(1),AR(2)  <-AR(0)～AR(7)の内容を表示します。
70      PRINT AR(3),AR(4),AR(5)
80      PRINT AR(6),AR(7)
90      PRINT
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     TIME 50
130     GOTO *LOOP
140     '
>RUN
 0 1 2
 3 4 5      ┐
 6 7        ┘①

 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
```

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

【SFTR】



【ＡＲ(ｎ)にはＷＡＩＴコマンドも使用できます。】

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

【Q】配列変数がＡＲ(０)～ＡＲ(３１)では足りない。
【A】ポイントデータ変数Ｘ(ｎ)、Ｙ(ｎ)、Ｕ(ｎ)、Ｚ(ｎ)も配列変数として演算に使用することができます。ｎはＰ版で０～３００ですがｎ＝０の時は現在座標を表す特殊な配列になります。これらにはＡＲ()にはあるローテート命令はありません。ＮＥＷＰコマンドで０に初期化されます。

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

```
80        U(J)=X(J)+Y(J)
90       PRINT U(J)
100     NEXT I
>RUN
 100 110 200
 101 111 201
 102 112 202
 103 113 203
 104 114 204
 105 115 205
 310
 312
 314
 316
 318
 320
```

【ＡＴＡＮ】　　機能：演算　　種別：コマンド　　サポート：Ｐ

■書　式　　ATAN n m v

n,m変数,定数 >0

v：結果が入る変数(v=atan(m/n))

■解　説　　n､mのアークタンジェントを求めます。

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

【ＡＸＩＳ】　　機能：パルス　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　　AXIS n

n：軸番号

1≦n≦4

1：X軸　2：Y軸　3：U軸　4：Z軸

■解　説　　パルス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:ｽﾍﾟｰｽｷｰ
     └PGﾎﾞｰﾄﾞ選択:TABｷｰ
```

ＭＯＤＥ　４での対応表

AXIS n	ﾊﾟﾙｽ出力	原点復帰ｾﾝｻ入力	座標関数
1	xcw/xccw	入力 16/入力 17	X(0)
2	ycw/yccw	入力 18/入力 19	Y(0)
3	ucw/uccw	入力 20/入力 21	U(0)
4	zcw/zccw	入力 22/入力 23	Z(0)

また、Ｚ版のＭＯＤＥ４では、ＭＯＶＥ、ＲＭＯＶ、ＨＯＭＥ、ＰＵＬＳＥの各パルス発生コマンドのサポート軸を選択します。

【ＢＫＣＮＴ】　機能：デバッグ　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　BKCNT n

n：ﾌﾞﾚｰｸまでの回数

1≦n≦128

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

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

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

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

【ＢＬ＿ＡＮＤ】　機能：Ｉ／Ｏ　種別：コマンド　サポート：Ｐ

■書　式　BL_AND A1,A2,n

A1,A2：入力ポート、メモリーＩ／Ｏ、変数、定数

n：出力ポート

※“＿”はアンダーバーです。

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

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

前記のプログラムをＩＦ文でかくとこんな感じ．．．

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

```
130     *ON1
140     ON 1
150     RETURN
160     *OFF1
170     OFF 1
180     RETURN
```

【ＢＬ＿ＯＲ】　　機能：Ｉ／Ｏ　　種別：コマンド　　サポート：Ｐ

■書　式　BL_OR A1,A2,n

A1,A2：入力ポート、メモリーＩ／Ｏ、変数、定数

n：出力ポート

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

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

ＩＦ文で書くとこんな感じ．．．

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

【ＢＲＫ】　　機能：デバッグ　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　BRK

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

【ＢＳＹ】　　機能：パルス　　種別：関数　　サポート：Ｐ

■書　式　BSY(n)

0≦n≦3

■解　説　ＭＰＧ－３０３の動作状態を数値で返します。引数のｎの値は次の意味を持ちます。

n=0：現在アクセス中のＭＰＧを指定する。タスク０ではＰＧコマンドによって選択されたＭＰＧ。タスク１～３ではＭＰＧ＃１、タスク４～７はＭＰＧ＃２、タスク８～１１はＭＰＧ＃３となります。

n=1､2､3:それぞれの番号に対応したＭＰＧを指定します。また、関数が返す値は次の意味を持ちます。

0：ＭＰＧ動作中。ただしＪＯＧコマンドをのぞく。

1：ＭＰＧ正常終了、コマンド待ち状態。

15：ＳＴＯＰ　１コマンドにより減速停止。

240：ＳＴＯＰ　２コマンドにより急停止。

3：ＯＶＲＵＮ設定によって停止。

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

【ＣＡＬＬ】　　機能：ユーザー定義　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　CALL &Haddress A1 A2

&Haddress ：プログラムが入っている番地
A1(入力)：プログラムに引き渡す数値(定数でも変数でも良い)
A2(出力)：得た結果を返す変数

■解　説　　CALL &H6F00 98 B1

前記の例では、マシン語プログラム(＄６Ｆ００番地)をＣＡＬＬし、数値９８を引き渡し、結果をＢ１に格納する事になります。ＣＡＬＬされた側のプログラムでは、ＢＣレジスタにＡ１の値が、ＤＥレジスタにＡ２(変数)のアドレスが設定されます。

― マシン語インターフェース ―

ＣＡＬＬ文で呼ばれるマシン語プログラムは、次の事に注意して使用します。

１．ＴＮＹＦＳＣから引き渡される値は、次の様になっています。
　　ＢＣ　レジスタ　　入力数
　　ＤＥ　レジスタ　　出力アドレス
２．スタックは、使用してもよいがリターンまでに元に戻す。
３．レジスタの内容は、ＩＸ，ＩＹレジスタ以外は自由に使用できます。

```
        ORG 06F00H
DUMMY:   LD  a,c
         LD  (DE),a
         INC DE
         LD  a,b
         LD  (DE),a
         RET
```

この例では、入力された数を指定の変数に返すのみです。ＴＮＹＦＳＣ　ＲＥＶ３．２２では6E80H～6FFFHまでがユーザ用として使用できます。(このアドレスはＲＥＶ、版によって違います。)
一度ダウンロードされれば、ＲＡＭ上に残りパワオンリセットによってもクリアされません。尚、ＲＥＶ－３．２２ｅ以前のものではパワオンリセット毎にデータがクリアされるので、注意してください。
使用可能アドレス・マシン語プログラムの作成・ダウンロード、ユーザーエリアなどについては弊社技術係までお問い合わせ下さい。

【CMND】　　機能：ＭＰＧ－３０１　　種別：コマンド　　サポート：Ｐ

■書　式　CMND code

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

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

```
CMND &H08     "定速連続駆動=>ｽﾋﾟｰﾄﾞは起動周波数(REG 3)
```

【ＣＮＦＧ#】　　機能：ＲＳ－２３２Ｃ　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　CNFG# A1,A2,A3,

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

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

ＩＮＰＵＴ#，ＰＲＩＮＴ#，ＰＵＴ#，ＧＥＴ#()，ＴＳＴ#()

通信フォーマットの設定です

Ａ１：モード設定

0	:NP7B1S	NP=ﾉﾝﾊﾟﾘﾃｨ
1	:NP7B2S	P =ﾊﾟﾘﾃｨ有り
2	:P7B1S	7B=7ﾋﾞｯﾄ
3	:P7B2S	8B=8ﾋﾞｯﾄ
4	:NP8B1S	1S=1ｽﾄｯﾌﾟﾋﾞｯﾄ
5	:NP8B2S	2S=2ｽﾄｯﾌﾟﾋﾞｯﾄ
6	:P8B1S	
7	:P8B2S	

Ａ２：パリティ選択

0	:偶ﾊﾟﾘﾃｨXON/XOFF 有
1	:奇ﾊﾟﾘﾃｨXON/XOFF 有
2	:偶ﾊﾟﾘﾃｨXON/XOFF 無
3	:奇ﾊﾟﾘﾃｨXON/XOFF 無

Ａ３：ボーレート選択

0	:禁止
1	:19.2K
2	:9600
3	:4800
4	:2400
5	:1200
6	:600
7	:禁止

◆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】　　機能：デバッグ　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　CNT

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

【ＣＯＮＴ】　　機能：タスク操作　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　CONT n[,m,l]

n,m,l：タスク番号

Ｐ版　1≦n≦11

Ｚ版　1≦n≦7

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

【ＣＳＷ】　　機能：Ｉ／Ｏ　　種別：関数　　サポート：Ｐ・Ｚ

■書　式　CSW(n)

n：入力ポート番号

0≦n≦255　　Ｉ／Ｏ

-128≦n≦-1　　メモリーＩ／Ｏ

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

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

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

【！ＣＳＷ】　　機能：Ｉ／Ｏ　　種別：関数　　サポート：Ｐ

■書　式　!CSW(n)

n：入力ポート番号

0≦n≦255　　Ｉ／Ｏ

-128≦n≦-1　　メモリーＩ／Ｏ

■解　説　入力ポートの読み取りですが入力ポートの値が変わるまで関数値は返しません。返す値は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。

【Ｄ４５】　　機能：パルス　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　D45 n

Ｐ版　n=0：Ｘ・Ｙ，Ｚ・Ｕいずれも直線補完

n=1：Ｘ・Ｙ，Ｚ・ＵいずれもＤ４５移動

n=2：Ｘ・ＹのみＤ４５移動、Ｚ・Ｕは直線補完

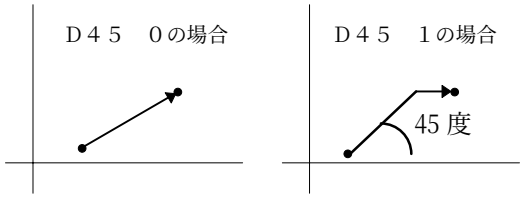
n=3：Ｚ・ＵのみＤ４５移動、Ｘ・Ｙは直線補完

Ｚ版　n=0：通常モード(直線補完)

n=1：４５度モード

■解　説　ＭＰＣのパルス発生では、直線補間パルス発生が基本となっています。この時ステップモータなどで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。Ｄ４

５はこの様な場合に移動方向を４５°と直線に分解して副軸の低パルスレートを避けます。この場合切り換え点で移動は一但減速停止し、再スタートとなります。



【ＤＡ】　　機能：ＭＩＦ－８１６ＡＤ　　種別：コマンド　　サポート：Ｐ

■書　式　　DA n
　　0≦n≦4095

■解　説　　ＭＩＦ－８１６ＡＤのＤ／Ａコンバータからの出力を変更します。Ｊ４－３１に出力されます。出力も１ｍＶ単位で、４．０９５Ｖまで出力出来ます。出力電圧は、ＡＤ(３)でモニタする事が出来ます。

```
DA 1000                    １Ｖ出力します
```

【ＤＥＣ】

FOR参照

【ＤＥＦ＿ＲＳＴ】　機能：Ｉ／Ｏ　　種別：コマンド　　サポート：Ｐ

■書　式　　DEF_RST A
　　A=&H00H～&HFFH　ビットパターン

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

```
10 DEF_RST &H01
20 *MAIN
```

【＃ＤＥＦＯ】　　機能：Ｉ／Ｏ　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　　#DEFO　文字列　番号

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

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

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

【＃ＤＥＦＳ】　機能：Ｉ／Ｏ　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　#DEFS　文字列　番号

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

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

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

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

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

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

【ＤＥＬＥＴＥ】　機能：編集　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　DELETE n,m

n,m：文番号(存在する文番号)

0<n,m≦32766

■解　説　削除する範囲を指定します。文番号ｎよりｍまでを削除。ある行より後を全て削除したい場合には、次の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コマンドで最終行の内容を待ち、これにより全消去を実施しています。

【＃ＤＩ】　機能：タスク操作　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　#DI

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

【E】　機能：ＲＳ－２３２Ｃ　種別：関数　サポート：Ｐ

■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。ＲＳ－ＩＳＯ等でノイズ源から分離して下さい。

【＃ＥＩ】　機能：タスク操作　種別：コマンド　サポート：Ｐ・Ｚ

■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。

【ＥＬＳＥ＿ＧＯＳＵＢ】　機能：制御文　種別：コマンド　サポート：Ｐ

■書　式　ELSE_GOSUB N
ELSE_THEN 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。

【ＥＬＳＥ＿ＴＨＥＮ】　機能：制御文　種別：コマンド　サポート：Ｐ

ELSE_GOSUB参照

【ＥＮＤ】　機能：制御文　種別：コマンド　サポート：Ｐ・Ｚ

■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】　機能：メンテナンス　種別：コマンド　Ｐ（3.30）Ｚ（2.30）以上

■書　式　ERASE

■解　説　フラッシュＲＯＭのプログラムをクリアーします。プログラム編集後のＲＵＮ、またはＦＷＲＩＴＥによりフラッシュＲＯＭに書き込まれたプログラムをクリアーします。初期化コマンド「ＭＰＣＩＮＩＴ」はフラッシュＲＯＭ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フラッシュＲＯＭに書き込まれたプログラムを展開します。全てのデータをクリアーする場合は「ＭＰＣＩＮＩＴ」「ＥＲＡＳＥ」を連続して実行します。

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

【ＥＲＲ＿ＯＮ】　機能：Ｉ／Ｏ　種別：コマンド　サポート：Ｐ

■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。また、この指定はリセット毎にクリアされてしまうので、プログラム中に記述しておく必要があります。また、エラー発生後エラー内容を確認するにはただちにターミナルを接続してＭＯＮコマンドを実行します。電源を入れ直して再実行したり、ＲＵＮをかけると、このデータは失われてしまいます。
◎ＥＲＲ＿ＯＮコマンド使用上の注意
ＥＲＲ＿ＯＮコマンドを用いた場合はタスク０をＥＮＤで停止しないで下さい。タスク０を停止すると監視プログラムが動作せず、エラー発生時に出力出来なくなります。

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

【ＦＥＤＤ】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

■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】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

■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】　　機能：パルス　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　FEDZ n
　　0≦n≦15

■解　説　疑似４軸ＰＧでＺ軸のスピードを定めます。ＸＹサーボ、Ｚ軸ステップモータ等の場合、ＸＹ軸とＺ軸では制御スピードが異なる為、このコマンドで予めスピードを設定します。ゲート・モーション(ＪＵＭＰ)での上昇、下降に有効です。

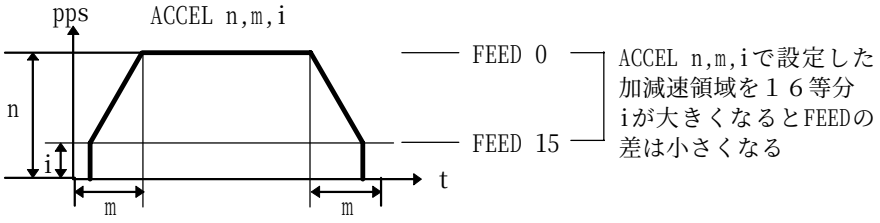
【ＦＥＥＤ】　　機能：パルス　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　FEED n
　　0≦n≦15　　0が最高速度
注)PG-1でのＦＥＥＤは６４段階です

■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。０～１５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　　Ｚ軸(Ｕ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      <-これは3秒ほどかかるので、ACCELｺﾏﾝﾄﾞはプログラムの
 SETPOS 0,0                   冒頭に設定して、実行中は変更しない。
*LOOP
 FOR I=0 TO 15
FEED I                     <-ｽﾋﾟｰﾄﾞを変えるにはFEEDを使う。FEEDは時間がかからない。
MOVE 100000,100000
TIME 50
MOVE 0,0
TIME 50
NEXT I
GOTO *LOOP
```

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

```
>PG 1
>MODE 6
>ACCEL 10000
```

```
>FEED 0
>FEDH 8
>RMOV 20000,0                <-2.5sec      HOMEしないとFEED 0のｽﾋﾟｰﾄﾞで動く
>HOME 0
>RMOV 20000,0                <-4.3sec      HOMEするとFEED 8で動く
>FEDH 0
>HOME 0
>RMOV 20000,0                <-2.5sec      FEDH 0のHOME後はFEED 0で動く
>FEED 15
>RMOV 20000,0                <-9.5sec
>FEDH 0
>HOME 0
>RMOV 20000,0                <-2.5sec      FEED 15でもFEDHが0ならｽﾋﾟｰﾄﾞはFEED 0

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

【ＦＩＮＤ#】　　機能：ＲＳ－２３２Ｃ　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　FIND# n

0≦n≦&H7F(アスキーコード)

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

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

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

【ＦＯＲ.. ＮＥＸＴ／ＤＥＣ】

機能：制御文　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　加算　FOR a1=a2 TO a3 ～ NEXT a1

a1：変数

a2,a3：変数、定数　a2<a3

減算(Ｐ版のみサポート)　FOR a1=a2 TO a3 ～ DEC a1

a1：変数

a2,a3：変数、定数　a2>a3

■解　説

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

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

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

【ＦＯＲＫ】　　機能：タスク操作　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　　FORK n,m

n：タスク番号

Ｐ版　1≦n≦11

Ｚ版　1≦n≦7

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            <-ﾗﾍﾞﾙ"*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 <-ﾀｽｸﾎﾟｰｽﾞ
?
 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
```

【ＦＲＥＥ】　機能：編集　種別：コマンド　サポート：Ｐ・Ｚ

■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
```

数値は版、Ｒｅｖによって異なる場合があります。

【ＦＷＲＩＴＥ】　機能：メンテナンス　種別：コマンド　Ｐ（3.30b）Ｚ（2.30b）以上

■書　式　FWRITE

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

```
>FWRITE
*++++++++                          ←フラッシュＲＯＭに書き込み中
>
```

【ＧＥＴ】　機能：ＲＳ－２３２Ｃ　種別：関数　サポート：Ｐ

■書　式　GET(0)

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

【ＧＥＴ＃】　機能：ＲＳ－２３２Ｃ　種別：関数　サポート：Ｐ・Ｚ

■書　式　GET#(0)

■解　説　ＲＳ－２３２Ｃ　ＣＨ１のバッファより１byte取り出して関数値をそのキャラクタのＡＳＣＩＩ値として与えます。
例)ＣＨ１に外部ターミナルを接続し、次のプログラムを実行しターミナルで Ａ 'のコードを出力(Ａキーを押す)すると次の様に表示されます。

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

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

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

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

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

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

＊ＣＮＦＧ＃でXON/OFFを有効にすると&H13、&H11はとれません。

＊ＲＳ－２３２Ｃのバッファは６４byteです。

◇Ｐ版でのＧＥＴ＃、ＧＥＴＮ＃の弱点

Ｐ版では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。Ｚ版ではこうした問題は発生しません。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のＰ版における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

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

```
 PRINT T1,RS(1),A8
 END

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

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

(MASTER2)
 *AG
  A=49
  CNFG# 4,0,2
  PUT# A
  C=10
 *LP
 GOSUB *GT
  C=C-1
  IF C>0 THEN *LP
  PRINT A
  TIME 10
  GOTO *AG
 *GT
  T1=100
 *GT1
  IF RS(1)>0 THEN *GT9
  T1=T1-1
  TIME 1
  IF T1>0 THEN *GT1
  GOTO *RSE
 *GT9
  A8=GET#(0)

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

【ＧＥＴＮ＃】　　機能：ＲＳ－２３２Ｃ　種別：コマンド　　サポート：Ｐ・Ｚ

■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#      ┐ ＧＥＴＮ＃の割込み禁止により
30 GETN# B0    ┘ シリアル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#      │ 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  ┘
```

【ＧＯＳＵＢ】　　機能：制御文　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　GOSUB n

n：文番号(存在する文番号)またはラベル

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

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

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

```
10     GOSUB 10
>RUN
 # 10
!! Stack Overflow
```

【ＧＯＴＯ】　　機能：制御文　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　GOTO n

n：文番号(存在する文番号)またはラベル

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

【ＨＣＳＷ】　機能：Ｉ／Ｏ　種別：関数　サポート：Ｐ・Ｚ

■書　式　HCSW(n)

ｎ：ポート番号

0≦n≦255

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

【ＨＯＭ】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　HOM

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

```
MOVZ 0
MOVE 0 0 0
```

を連続して実行し、Ｚ軸上昇、Ｘ，Ｙ，Ｕ移動というシーケンスとなっています。

【ＨＯＭＥ】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　Ｘ，Ｙ軸原点復帰

HOME n,x,y

ｎ：XS1～YS2までのセンサーパターン

x,y：退避移動量

Ｚ，Ｕ軸原点復帰

HOMZ n,z,u

ｎ：US1～ZS2までのセンサーパターン

z,u：退避移動量

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

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

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。　参照コマンド：ＳＨＯＭ

```
10 MODE 5                    <-MODE設定は一番始め
20 OVRUN 0                   <-ｵｰﾊﾞｰﾗﾝ設定　この場合はｵﾊﾞｰﾗﾝ停止無し
30 ACCEL 30000               <-加減速ﾃｰﾌﾞﾙ作成
40 SHOM 1,0,100              <-原点復帰設定　X軸CW方向,Y軸は無し,速度353pps
50 FEDH 15                   <-退避移動ｽﾋﾟｰﾄﾞ設定 0(高速)～15(低速)
60 HOME 1,-1000,0            <-原点復帰 XS1がONまで,退避量X=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ﾊﾟﾙｽでX,Y軸の原点復帰実行
                                 <-<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
```

☆ＨＯＭＺコマンドの実行後の座標値

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

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 <-確認
>SHMZ 1,0,1000       <-原点復帰ﾊﾟﾀｰﾝ U:CW Z:無し ｽﾋﾟｰﾄﾞ:1000
>HOMZ 1,0,-200       <-原点復帰 ｾﾝｻｰﾊﾟﾀｰﾝ:US1=ON 退避移動量:Z=0,U=-200
PG 1#1(X,Y,Z,U) -500 -500 0 0 [XYZ,U] 100 100 <-実行後の座標
```

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       <-原点復帰ﾊﾟﾀｰﾝ U:無し Z:CW ｽﾋﾟｰﾄﾞ:1000
>HOMZ 4,-200,0       <-原点復帰 ｾﾝｻｰﾊﾟﾀｰﾝ:ZS1=ON 退避移動量:Z=-200,U=0
PG 1#1(X,Y,Z,U) 0 0 0 0 [XYZ,U] 100 100 <-実行後の座標
```

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       <-原点復帰ﾊﾟﾀｰﾝ U:CW Z:CW ｽﾋﾟｰﾄﾞ:1000
>HOMZ 5,-200,-200    <-原点復帰 ｾﾝｻｰﾊﾟﾀｰﾝ:US1,ZS1=ON 退避移動量:Z=-200,U=-200
PG 1#1(X,Y,Z,U) -500 -500 0 0 [XYZ,U] 100 100 <-実行後の座標
```

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

■省略可能なパラメーターについて
ＨＯＭＥコマンドの第２(Ｘ軸退避量)、第３(Ｙ軸退避量)パラメーターは省略可能ですが、その場合は以前に設定された数値が有効になります。プログラム中で省略してしまうと、そのうちにどの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コマンドに限らず省略可能なパラメーターでもプログラムには全部記述しておきましょう。

```
HOME 1,500,500      <-ﾀﾞｲﾚｸﾄｺﾏﾝﾄﾞで設定した値が有効になる
>100 HOME 1         <-ﾌﾟﾛｸﾞﾗﾑ中に書いておかないとそのうちわからなくなる。
                      ﾎﾞｰﾄﾞを交換したりRAM化けを起こしたら期待通りに動作しなくなる。
```

【Ｚ版について】

■書　式　　HOME n[,x,y]

n：論理状態(原点復帰のセンサ・パターン)

0≦n≦&H3F

x,y：退避移動量(デフォルト２００パルス)

-32767≦x,y≦32767

■解　説　　ｘ，ｙは原点復帰に先立つ原点エリアよりの退避移動量で通常２００パルスとなっています。このスピードはＦＥＤＨによって設定されます。又リセットにて８となります。

BIT0	BIT1	BIT2	BIT3	BIT4	BIT5
IN16	IN17	IN18	IN19	IN20	IN21
XS1	XS2	YS1	YS2	US1	US2

例えば、原点復帰センサーをX用にＩＮ１７、YにＩＮ１９を割あてたとします。そして、原点でセンサーが共にＯＮとなるとするとビット・パターンは、次の図の様になり、コマンドとしては、”ＨＯＭＥ　＆Ｈ０Ａ”として使用します。

BIT0	BIT1	BIT2	BIT3
IN16	IN17	IN18	IN19
0	1	0	1

⇒1010=&H0A

ＭＯＤＥ　４　でのＨＯＭＥコマンドでは入力は２ｂｉｔごとに意味を持ちＡＸＩＳのコマンドによって次の様に変わります。例えば、同じ　ＨＯＭＥ　３　でもＡＸＩＳが３であればU軸の原点復帰となりＩＮ２０、ＩＮ２１が有効となるわけです。ＨＯＭＥのスピードはＳＨＯＭで指定されます。

AXIS n	ﾊﾟﾙｽ出力	原点復帰ｾﾝｻ入力	座標関数
1	xcw/xccw	IN16/IN17	X(0)
2	ycw/yccw	IN18/IN19	Y(0)
3	ucw/uccw	IN20/IN21	U(0)
4	zcw/zccw	IN22/IN23	Z(0)

■Ｚ版のＭＯＤＥ　１の原点復帰について

ＭＯＤＥ　１でのＨＯＭＥコマンドはＸ，Ｙ，Ｕの３軸同時移動です。Ｘ，Ｙ軸の原点復帰方向とＸ，Ｙ，Ｕ，Ｚ軸のスピードはＳＨＯＭコマンドで設定できますが、Ｚ，Ｕ軸の原点復帰方向はＣＣＷ方向に固定されていて変更出来ません。Ｘ，Ｙ軸の原点復帰時の退避移動量はＨＯＭＥコマンドで設定しますがＺ，Ｕ軸の退避移動量はＳＨＭＺＵコマンドで設定します。Ｚ，Ｕ軸の原点復帰方向を変更する時はパルスの配線を入れ替えます。

```
MODE 1
SHOM 1,4,400                <-X,Y軸CW方向に原点復帰、ｽﾋﾟｰﾄﾞ186Hz
SHMZU 100,100               <-Z,U軸退避移動量　CW方向に100ﾊﾟﾙｽ
HOME &H15,-200,-200         <-X,Y=CW、U=CCW原点復帰、X,Yの退避移動はCCW200ﾊﾟﾙｽ
HOMZ 1                      <-Z=CCW原点復帰
```

■Ｚ版の原点センサーポートについて

ＭＩＦ－８１６からパルス出力をするＺ版のＨＯＭＥコマンドでの原点センサーは入力１６～２１に割当られています。ＨＯＭＥコマンドは指定されたパターンと入力パターンが一致するまでパルスを出し続けます。このポートには原点センサー以外のＳＷは接続しないで下さい。原点センサー以外のＳＷを接続するとＨＯＭＥが正常にできなくなります。たとえば入力１８に原点センサー以外のＳＷが接続されていてＯＮなっていたとするとＨＯＭＥ　＆Ｈ０Ａでは原点復帰ができません。(MODE 4を除く)

【ＨＯＭＺ】　　機能：パルス　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　HOMZ n[,z,u]

n：US1～ZS2までのセンサーパターン

z,u：退避移動量

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

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

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

```
HOMZ ５ ０ ０
```

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

```
HOME ０ ０ ０
```

原点復帰の方向と速度を決めるのはＳＨＭＺコマンドです。

参照コマンド：ＳＨＭＺ，ＨＯＭＥ

【Ｚ版について】

■書　式　HOMZ n

n：原点センサビット・パターン

0≦n≦3

■解　説

BIT0	BIT1
IN22	IN23
ZS1	ZS2

ＨＯＭＺのスピードは、ＨＯＭＥと同様ＳＨＯＭで指定されます。

【ＨＰＴ】　　機能：Ｉ／Ｏ　　種別：関数　　サポート：Ｐ

■書　式　HPT(n)

0≦n≦8

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

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

```
PRINT HPT(1),HPT(2)
 １ ０
```

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

【ＨＳＷ】　　機能：Ｉ／Ｏ　　種別：関数　　サポート：Ｐ・Ｚ

■書　式　HSW(n)

n：ポートナンバー

0≦n≦255　　Ｉ／Ｏ

-128≦n≦-1　メモリーＩ／Ｏ

0～255

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

【！ＨＳＷ】　　機能：Ｉ／Ｏ　　種別：関数　　サポート：Ｐ・Ｚ

■書　式　!HSW(n)

0≦n≦255　　Ｉ／Ｏ

-128≦n≦-1　メモリーＩ／Ｏ

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

【ＨＷＳ０】　　機能：Ｉ／Ｏ　　種別：関数　　サポート：Ｐ・Ｚ

■書　式　HWS0(n)
HWS1(n)

n：ポート番号

0≦n≦255　　Ｉ／Ｏ

-128≦n≦-1　メモリーＩ／Ｏ

■解　説　タイム・アウト付き、ウェイト関数。インタロックとしては、次のコマンドと同様です。

ＨＷＳ０(ｎ)　～　⇔　ＷＡＩＴ　ＨＳＷ(ｎ)＝０

ＨＷＳ１(ｎ)　～　⇔　ＷＡＩＴ　ＨＳＷ(ｎ)＝１

ＨＷＳ０(ｎ)，ＨＷＳ１(ｎ)は関数として用います。タイム・アウト機能が追加されており、タイム・アウト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(時間制限)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。

【HWS１】　　機能：Ｉ／Ｏ　　種別：関数　　サポート：Ｐ・Ｚ

HWS0参照

【ＩＦ～ＴＨＥＮ／ＧＯＳＵＢ】

機能：制御文　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　IF～条件式～THEN/GOSUB n

n：文番号 1≦n≦32766またはラベル

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

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

Ｐ版ではＥＬＳＥが使用できます。

【ＩＮ】　　機能：Ｉ／Ｏ　　種別：関数　　サポート：Ｐ・Ｚ

■書　式　IN(n)

n：ﾊﾞﾝｸﾅﾝﾊﾞｰ

0≦n≦31　　Ｉ／Ｏ

-16≦n≦-1　メモリーＩ／Ｏ

8255ｱﾄﾞﾚｽ

■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。次にバンクナンバーと入力ポートの関係を一部表に示します。

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
```

■８２５５のアドレス指定

IN(n)のnに８２５５のアドレスを与えて入力するもできます。この場合も5msecのタイマーは入りません。出力の状態も読み取ることができます。ただし、データは反転してます。

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

【ＩＮＰＵＴ】　機能：ＲＳ－２３２Ｃ　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　INPUT A1[,A2,A3]

A1～A3：変数

■解　説　ＲＳ－２３２Ｃ　ＣＨ０(プログラムポート)からデータを入力します。

下記の様に、プログラム実行中にプログラムに数を引き渡すのに使用します。変数は１～３個で複数指定の場合はスペースで区切り、続けて入力します。

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

【ＩＮＰＵＴ＃】　機能：ＲＳ－２３２Ｃ　種別：コマンド　サポート：Ｐ・Ｚ

■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。

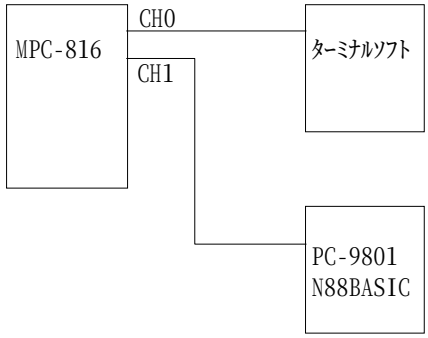
受信処理は、数字、文字列とデリミタ・ターミネーターに限られ次の書式となります。(デリミタは「スペース」(&H20)、ターミネーターは「ＣＲ」(&H0D)です。)

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。

【ＩＯ＿ＣＮＴ】　機能：ソフトカウンタ　種別：コマンド　サポート：Ｐ

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

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

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

【ＪＭＰＺ】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

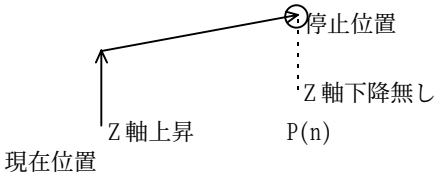
■書　式　JMPZ P(n)

n：点番号

Ｐ版　1≦n≦300

Ｚ版　1≦n≦255

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



【ＪＯＧ】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　Ｐ版について

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(ＨＥＸで３Ｅ８)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(ＨＥＸで３００)しか動きません。

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

◆動作範囲のＭＩＮが０の場合

次のプログラム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
```

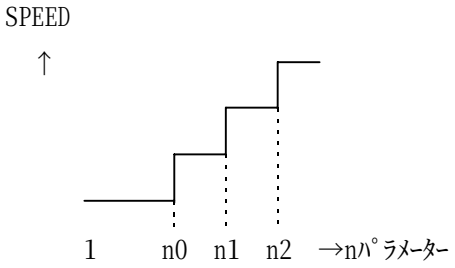
■ＪＯＧの設定範囲について。
ＭＯＤＥ　６でのＪＯＧ　ｎ、ｍのｎの設定範囲は、最大５０００まで、ＡＣＣＥＬ　ｎ，ｍ，ｉの加速距離m以下です。ＡＣＣＥＬ設定時にm省略するとｎの１／５までが設定範囲になります。

```
>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
IWATSU SC-7201 UNIVERSAL COUNTER



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

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。

【Z版について】

■書　式　　JOG n,A1

n：加速距離　1≦n≦加速距離(ＡＣＣＥＬ指定)
A1：入力ポート指定　省略時有効入力はSW(4)～SW(7)

JOGZU n,A1

n：加速距離　1≦n≦加速距離(ＡＣＣＥＬ指定)
A1：入力ポート指定　省略時有効入力はSW(0)～SW(3)

■解　説　　ＪＯＧコマンドはボタンを押した状態で連続移動する場合に用いられるティーチング機能です。ｎは加速距離で、これはＡＣＣＥＬで指定した加速距離より小さい値を設定します。その範囲での数値が大き

いほどスピードは早くなります。Ａ１は入力ポートバンク指定ですが省略すると　&Ｈ４８(ＭＰＣ－８１６の入力０～７)として扱われます。ＪＯＧがＭＳＢ側、ＪＯＧＺＵがＬＳＢ側にアサインされ、それぞれ独立にアドレスを設定することができます。ＪＯＧ、ＪＯＧＺＵでの移動範囲はＸＲＡＮＧ～ＵＲＡＮＧで設定できます。

ビット	B7	B6	B5	B4	B3	B2	B1	B0
A1 省略時入力	SW(7)	SW(6)	SW(5)	SW(4)	SW(3)	SW(2)	SW(1)	SW(0)
制御軸	YCCW	YCW	XCCW	XCW	ZCCW	ZCW	UCCW	UCW

サンプル①

```
10      CHGREV 22
20      XRANG 10000,0
30      YRANG 10000,0
40      URANG 10000,0
50      ZRANG 10000,0
60      *LOOP
70      IF SW(0)=1 GOSUB *JOGZU  <-SW(0)がONしている間はUCW に動く
80      IF SW(1)=1 GOSUB *JOGZU  <-SW(1)がONしている間はUCCWに動く
90      IF SW(2)=1 GOSUB *JOGZU  <-SW(2)がONしている間はZCW に動く
100     IF SW(3)=1 GOSUB *JOGZU  <-SW(3)がONしている間はZCCWに動く
110     IF SW(4)=1 GOSUB *JOGXY  <-SW(4)がONしている間はXCW に動く
120     IF SW(5)=1 GOSUB *JOGXY  <-SW(5)がONしている間はXCCWに動く
130     IF SW(6)=1 GOSUB *JOGXY  <-SW(6)がONしている間はYCW に動く
140     IF SW(7)=1 GOSUB *JOGXY  <-SW(7)がONしている間はYCCWに動く
150     GOTO *LOOP
160     *JOGZU
170     JOGZU 500
180     RETURN
190     *JOGXY
200     JOG 500
210     RETURN
```

サンプル②　　①のプログラムをＩＮ()に替えたもの。

```
10      CHGREV 22
20      XRANG 10000,0
30      YRANG 10000,0
40      URANG 10000,0
50      ZRANG 10000,0
60      *LOOP
70       A=IN(0)
80       A1=A^&H000F
90       A2=A^&H00F0
100     IF A1<>0 GOSUB *JOGZU
110     IF A2<>0 GOSUB *JOGXY
120     GOTO *LOOP
130     *JOGZU
140     JOGZU 500
150     RETURN
160     *JOGXY
170     JOG 500
180     RETURN
```

サンプル③　　②の入力ポートを変更してみました。入力はSW(8)～SW(15)になります。

```
10      CHGREV 22
20      XRANG 10000,0
30      YRANG 10000,0
40      URANG 10000,0
50      ZRANG 10000,0
60      *LOOP
70       A=IN(1)
80       A1=A^&H000F
90       A2=A^&H00F0
100     IF A1<>0 GOSUB *JOGZU
110     IF A2<>0 GOSUB *JOGXY
120     GOTO *LOOP
130     *JOGZU
140     JOGZU 500,&H004A
```

```
150     RETURN
160     *JOGXY
170     JOG 500,&H004A
180     RETURN
```

【Ｑ１】ＪＯＧコマンドは無限のパルス出力が可能か。
【Ａ１】ＪＯＧコマンドはＸＲＡＮＧ等のＲＡＮＧコマンドで動作範囲の制限を受けるので無限のパルス発生はできません。

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

ＸＲＡＮＧで＋－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。
```

【ＪＯＧＺＵ】　機能：パルス　種別：コマンド　サポート：Ｚ

JOG参照

【ＪＵＭＰ】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　JUMP P(n)[,L]

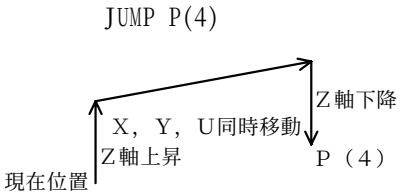
n：点番号

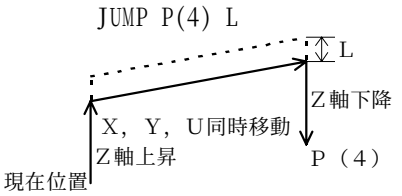
L：Ｚの上限値、L=0であれば最上端まで上昇

Ｐ版　1≦n≦300
　　0≦L≦8388607

Ｚ版　1≦n≦200
　　0≦L≦32766

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





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

【ＫＥＹ】　　機能：メンテナンス　　種別：コマンド　　サポート：Ｐ

■書　式　KEY n

n：ロック番号

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

KEY 9812

とコマンドを実行すると9812をパスワードとしてロックがかかり、フラッシュＲＯＭ上のデータは変更することができなくなります。パスワートは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。

【ＬＥＴ】　　機能：演算　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　LET ～算術式～　　（ＬＥＴは省略できます。）

■解　説　ＴＮＹＦＳＣでサポートされているのは、次の２項整数演算のみです。

A1=A2+A3　加
A1=A2-A3　減
A1=A2*A3　乗
A1=A2/A3　除
A1=A2%A3　余
A1=A2^A3　論理積（ＡＮＤ）
A1=A2|A3　論理和（ＯＲ）
A1=A2xA3　排他的論理和（ＸＯＲ）(xはｽﾓｰﾙｴｯｸｽ)

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。Ｘはスモールｘです。Ｐ版では３byte整数、Ｚ版では２byte整数での演算となります。

【ＬＩＳＴ】　　機能：編集　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　LIST [n,m]

n：開始文番号或は開始＊ラベル
m：表示行数
0≦n≦32766
1≦m≦127

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

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

【ＬＯＣ】　　機能：ＬＣＤ　　種別：コマンド　　サポート：Ｐ

■書　式　　LOC n,m[,s]

n：行(縦)1～2または4

m：列(横) 1～16または20

s：カーソル

s=15：カーソル＋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ＣＬＤ機種指定と画面消去になります。

ＣＬＤ機種(n=0)

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

ＬＣＤ関係のコマンド：LOC,PRC,PRD,PRS

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

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

【ＬＰ】　　機能：ファイル　　種別：コマンド　　サポート：Ｐ

■書　式　　LP n

0≦n≦3

■解　説　　点データの取り出し。Ｐ版では点データを通常使用しないメモリエリアに保存する事ができます。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。

【ＬＳＣＮＴ】　　機能：編集　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　　LSCNT [n]

n：表示行数

1≦n≦127

■解　説　　標準リスト行数の設定、表示ｎを省略すると現在の設定行数を表示

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

これは１画面で表示できる行数です。ＬＩＳＴは小さな画面でも使用し易い様にＬＳＣＮＴで行数管理されています。次を続いて表示するにはＬＩＳＴ　↓とします。最初から表示する場合は、ＬＩＳＴ０↓とします。デフォルトは２０です。（↓はリターンキー）

【M】　機能：演算　種別：配列　サポート：Ｐ

■書　式　M(n)

■解　説

M(0)～M(1199)
M(1200)～M(2399)　点ﾃﾞｰﾀ#0と共通
M(2400)～M(3599)　点ﾃﾞｰﾀ#1と共通
M(3600)～M(4799)　点ﾃﾞｰﾀ#2と共通
M(4800)～M(5999)　点ﾃﾞｰﾀ#3と共通

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

【ＭＩＯ】　機能：Ｉ／Ｏ　種別：コマンド　サポート：Ｐ

■書　式　MIO n

n：816 or 240

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

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

※()の中の数値はＭＩＯ－２４８の８点の入力ポート番号

【ＭＯＤＥ】　　機能：パルス　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　　MODE n

■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
```

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

ＭＯＤＥ　５：最大パルスレート２９．０kppsＸＹ及びＺＵの２軸ずつの直線補間パルス発生機能を持ちます。パルス幅が２０μsec確保されておりステップモータ用ドライバ向けの使用となっています。

ＭＯＤＥ　６：最大パルスレート５６．３kppsＸＹ及びＺＵの２軸ずつの直線補間パルス発生機能を備えています。パルス幅が４．２μsecの為ステップモータドライバには不適です。

	ﾓｰﾄﾞ	パルス幅	デュティ比	最大ﾊﾟﾙｽﾚｰﾄ
Ｐ	5	20 μ sec	50%以上(*)	29.0kpps
	6	4.2 μ sec	50%以下	56.3kpps

(*)但し、ＡＣＣＥＬ　２５０００以下の場合

【Ｚ版について】

■書　式　　MODE n

■解　説　　ＭＯＤＥコマンドは、ＰＧの種類を選択します。用途によって１～４までの値を設定します。ＭＯＤＥ２が当社推奨モードです。

ＭＯＤＥ　１：ＸＹＵ３軸及びＺ軸のパルス発生機能が得られます。注意すべき事は、ＯＶＲＵＮを指定した場合の最大スピードが２５kppsという事のみです。

　用　途　　：サーボ・コントローラによる４軸直交ロボット等

ＭＯＤＥ　２：パルスモータを主体とした低速系のパルス発生機能です。Ｘ、Ｙの２軸とＺ軸の３軸仕様となっています。

　用　途　　：ステップモータ（４相、２相の制御）

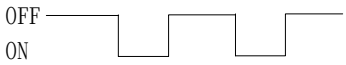
ＭＯＤＥ　３：サーボモーターを主体とした高速系のパルス発生機能です。Ｘ、Ｙの２軸とＺ軸の３軸で４７kppsまで動作します。

　用　途　　：サーボモータの制御

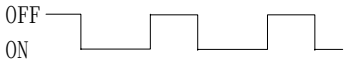
ＭＯＤＥ　４：ステップモーターを主とした単軸複合動作のモードです。ＡＸＩＳ命令で軸を切り換えることができます。

	ﾓｰﾄﾞ	ｾﾝｽ停止	最大ﾊﾟﾙｽﾚｰﾄ	最小ﾚｰﾄ	最大移動距離	パルス幅
Ｚ	1	Ｚ軸のみ	31.6kpps	400	32766	8 μ sec
	2	X,Y,Z	33.6kpps	200	65535	15 μ sec
	3	X,Y,Z	62.0kpps	2K	65535	8 μ以上
	4	X,Y,Z,U	31.2kpps	400	65535	10 μ sec

■ＭＯＤＥ　３で注意していただきたいのは低速時にパルスのデューティーが反転するということです。つまり通常ではパルスがＯＮしている時間よりＯＦＦしている時間が短いのですが、これが反対にＯＮしている時間の方が長くなります。ドライバーによってはデューティーが５０％以下と規定しているものがあります。



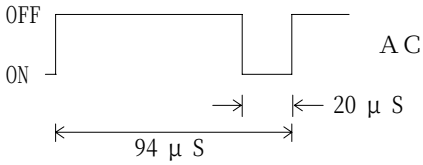
高速時



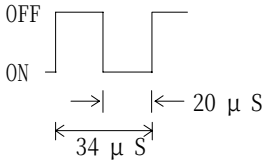
低速時 ﾃﾞｭｰﾃｨｰが反転する

■モード別パルス出力状況実測

【ＭＯＤＥ　５】　　ＯＮ時間が約２０μsecで一定

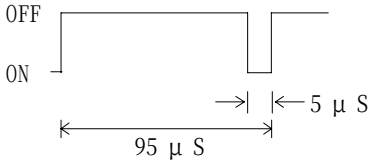


ＡＣＣＥＬ　１００００

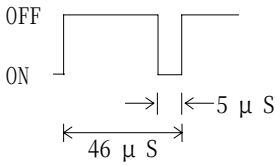


ＡＣＣＥＬ　３００００

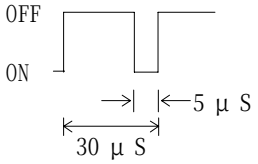
【ＭＯＤＥ　６】　　ＯＮ時間が約５μsecで一定



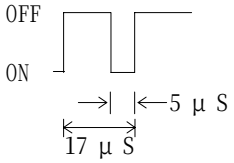
ＡＣＣＥＬ　１００００



ＡＣＣＥＬ　２００００

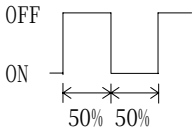


ＡＣＣＥＬ　３００００

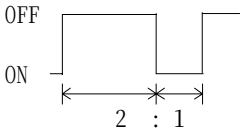


ＡＣＣＥＬ　５００００

【ＨＯＭＥコマンド】ＨＯＭＥコマンドはＭＯＤＥ、スピードに関係無くデューティー５０％



【ＪＯＧコマンド】　ＪＯＧコマンドはＭＯＤＥ、スピードに関係無く２：１



【ＭＯＮ】　　機能：デバッグ　　種別：コマンド　　サポート：Ｐ・Ｚ

■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、ＦＴＭでは＜ＣＴＲＬ＞＋＜Ｍ＞によって対応する文番号のコマンドを表示します。

【ＭＯＶＥ】　　機能：パルス　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　　MOVE X,Y
MOVE P(n)

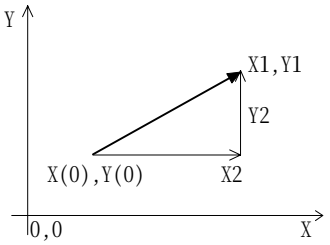
■解　説　　絶対座標移動です。ＸＹ軸についてＸ、Ｙの位置までパルス出力します。ＡＣＣＥＬによって最大スピードが決まりＦＥＥＤコマンドでスピードを切り換える事ができます。ＭＯＶＥ　Ｘ１、Ｙ１の場合のパルス発生量Ｘ２、Ｙ２は

```
X2=X1-X(0)
Y2=Y1-Y(0)
```

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

```
MOVE X(n),Y(n)
```

固定量パルスを出力する場合は、ＲＭＯＶを使用して下さい。
途中で停止する場合はＳＴＯＰコマンドを使用します。
※パルス発生にはＭＯＶＥ５とＭＯＤＥ６がありますが、ＭＯＤＥコマンドはいずれのモードでも同様にに使用できます。異なった動作をします。選択についてはＭＯＤＥコマンドを参照して下さい。



【Ｚ版について】

■解　説　　絶対座標移動(座標系で管理された位置への移動)パルス発生です。スピード変更はＦＥＥＤコマンドです。パルス発生にはＭＯＤＥ１～ＭＯＤＥ４までありそれぞれに特性の異なった動作をします。ＭＯＤＥ選択についてはＭＯＤＥコマンドを参照して下さい。

ＭＯＤＥ１

■書　式　　MOVE X,Y,U
MOVE P(n)
X,Y,U：各座標値　　-32767≦X,Y,U≦32767

n：点番号

ＭＯＤＥ２、ＭＯＤＥ３

■書　式　MOVE X,Y,J

MOVE X(n),Y(n),J

X,Y：各座標値　　-32767≦X,Y≦32767

n：点番号

Ｊ：停止条件。停止条件を与えない場合はＪ＝０。

ＭＯＤＥ２、ＭＯＤＥ３はX，Ｙの２軸移動です。３番目の引数は停止条件で移動中に入力条件によって停止する事ができます。

ＭＯＤＥ４

■書　式　MOVE X,J

MOVE X(n),J

X：各座標値　　-32767≦X≦32767

Ｊ：停止条件

１軸の移動コマンドです。出力軸はX～Ｚを選択できます。ＭＯＤＥ４はＡＸＩＳコマンドで出力軸を選択します。

ＡＸＩＳ　１にて　パルス出力ポートは　Ｘ軸
ＡＸＩＳ　２にて　パルス出力ポートは　Ｙ軸
ＡＸＩＳ　３にて　パルス出力ポートは　Ｕ軸
ＡＸＩＳ　４にて　パルス出力ポートは　Ｚ軸

となります。停止条件がＪが不要の場合はＪを０として下さい

※停止条件は、ＳＷ(２４)～ＳＷ(３１)までのビットパタンと停止モードです。停止モードの選択は上位８bit、入力ビットパタンは下位８bitで設定します。

Ｊ＝＆Ｈ××△△

××＝０　減速停止

××≠０　急停止

△△の表

ビット	Ｂ７	Ｂ６	Ｂ５	Ｂ４	Ｂ３	Ｂ２	Ｂ１	Ｂ０
入力ポート	３１	３０	２９	２８	２７	２６	２５	２４

停止条件はビットによって指定されたポートの論理和となります。例えば、Ｊ＝３とします。この場合ポート２４、ポート２５のどちらかがＯＮとなればパルスの出力は停止されます。なお、停止条件Ｊの使用方法はＲＭＯＶ、ＭＯＶＺ、ＲＭＶＺのいずれとも共通です。

＊注意　　ＭＯＤＥ　２のＭＯＶＥ、ＲＭＯＶコマンドの条件停止について

ＭＯＶＥ　ｘ，ｙ，ＪのＪは途中停止の入力パターンです。プログラムには省略せずに記述して下さい。また、この条件はＰ(ｎ)指定やＳＴＰＺＵ　０、０、０でクリアーされます。ポイント番号を指定した移動で途中停止するときはＰ(ｎ)をＸ(ｎ)，Ｙ(ｎ)に分けて与えます。

```
10      CHGREV 22
20      MODE 2
30      ACCEL 3000
40      FEED 0
50      SETP 1,10000,10000
60      SETPOS 0,0
70      MOVE X(1),Y(1),&H0001    <--ﾎﾟｲﾝﾄ番号指定の停止条件付きﾊﾟﾙｽ発生
80      PRINT 1,X(0),Y(0)        <--現在点表示
90      MOVE P(1)
100     PRINT 2,X(0),Y(0)
110     MOVE 0,0
120     PRINT 3,X(0),Y(0)
>RUN
 1 2405 2405              <--MOVE中にSW(24)がｵﾝになって途中停止しました。
 2 10000 10000            <--P(n)指定では停止条件が解除されます。(SW(24)はｵﾝのまま)
 3 0 0                    <--P(n)指定後は停止条件が解除されます。(SW(24)はｵﾝのまま)
```

```
>SETPOS 0,0
>MOVE 10000,10000,&H01    <--途中停止します。
>MOVE 0,0,0               <--第3ﾊﾟﾗﾒｰﾀｰが0で停止条件解除
>

SETPOS 0,0
>MOVE 10000,10000,&H01
>STPZU 0,0,0              <--STPZUによる停止条件のｸﾘｱｰ
>MOVE 0,0
```

【ＭＯＶＺ】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

ＭＯＤＥ５、ＭＯＤＥ６ともコマンド上の扱いは全く同等です。ＭＯＶＺはＭＯＶＥコマンドがＸＹ軸に対して有効なのに対してＺＵ軸に有効になります。ＡＣＣＥＬによって最大スピードが決まりＦＥＤＺコマンドでスピードを切り換える事ができます。

■書　式　MOVZ Z,U
MOVZ P(n)

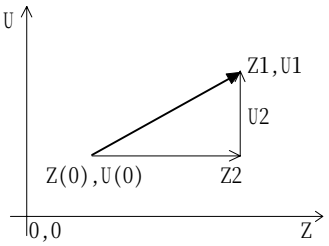
■解　説　絶対座標移動ＺＵ軸ついて、Ｚ，Ｕの位置までパルス出力します。ＭＯＶＺ　Ｚ１，Ｙ１の場合のパルス発生量Ｚ２，Ｙ２は

Z2=Z1-Z(0)
U2=U2-U(0)

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

MOVZ Z(n),U(n)

固定量パルスを出力する場合は、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。



【Ｚ版について】

■書　式　MOVZ Z,J
MOVZ Z(n),J
Ｚ：座標値　　-32767≦Z≦32767
Ｊ：停止条件

■解　説　ＭＯＶＺはＭＯＤＥ　１～ＭＯＤＥ　３に対して有効で、ＭＯＤＥ　４では無視されます。ＭＯＶＺはＺ軸に対する移動命令です。ＭＯＶＥと同様現在位置と目的地データの差分のみパルスが出力されます。尚、ＭＯＶＺコマンドはＦＥＤＺによってスピード設定されます。停止条件ＪはＭＯＶＥと同じです。スピードの変更はＦＥＤＺです。

【ＭＰＣＩＮＩＴ】　機能：メンテナンス　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　MPCINIT　（ダイレクトコマンドでの使用のみ）

■解　説　Ｐ／Ｚ版変更、システムアップデート後などにＭＰＣＩＮＩＴを実行しＲＡＭを初期化します。ＭＰＣＩＮＩＴが正しく実行されていないとプログラムがうまく入力出来なかったり、動作が異常になったりします。また、装置の調整にとりかかる前にＭＰＣＩＮＩＴを実行すると工場出荷状態にもどります、組立配線時に静電気などでＲＡＭの内容を壊してしまうことがあるため励行して下さい。プログラムには記述できません。
注意：ＭＰＣＩＮＩＴを実行すると点データやプログラムが壊されてしまいます。実行前にプログラムや点データはパソコンに保存して下さい。ＭＰＣＩＮＩＴはフラッシュＲＯＭに書き込まれたプログラムは消去しません。フラッシュＲＯＭの消去はＥＲＡＳＥコマンドです。

【Z版について】

Z版でMPCINITを実行する時、原点センサーポート(MIF-816の入力16～23)のいずれかがON状態にあるとMPCINITを終了しません。原点センサーにB接点タイプを使用していたり、装置が原点位置にある場合は注意して下さい。MIF-816のI／Oコネクターを抜いてからMPCINITを実行すると確実です。また、パルスが発生することもありますのでMIFのJ5コネクターも抜いてください。

【MTRX】　機能：パルス　種別：コマンド　サポート：P・Z

■書　式　MTRX m,n

m,n：パレットの縦、横の数

1≦m,n≦32766

■解　説　mが点i～j、nが点i～kへの分割量 （PALET参照）

【MTRX1】　機能：パルス　種別：コマンド　サポート：P

■書　式　MTRX1 m,n

m,n：パレットの縦、横の数

1≦m,n≦32766

■解　説　mが点i～j、nが点i～kへの分割量 （PALET参照）

【NEW】　機能：編集　種別：コマンド　サポート：P・Z

■書　式　NEW

■解　説　プログラムデータを全てクリアする。新たにプログラムを作成する場合に使用。ポイントデータの初期化はNEWPです。FREEで表示されるNEW後のステップ数

P版　2044

Z版　2044

【NEWP】　機能：編集　種別：コマンド　サポート：P・Z

■書　式　NEWP

■解　説　点データX(n)，Y(n)，Z(n)，U(n)が全て0となる。ROMの交換時にNEWPを実行して下さい。点データエリアはP版で300、Z版では255点です。

【NEXT】

FOR参照

【O_IN】　機能：I／O　種別：コマンド　サポート：P

■書　式　O_IN(n)

n：ポートバンク

0≦n≦31

■解　説　出力の状態をバンク単位で取得します。

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

【O＿SW】　機能：Ｉ／Ｏ　種別：関数　サポート：Ｐ

■書　式　O_SW(n)

n：出力ポート番号

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

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

【ＯＦＦ】　機能：Ｉ／Ｏ　種別：コマンド　サポート：Ｐ・Ｚ

ON参照

【ＯＦＦ＿ＡＮＤ】　機能：Ｉ／Ｏ　種別：コマンド　サポート：Ｐ

■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されます
>PR O_SW(0)
 0                          <-ｵﾌのままです
>ON 0                       <-出力0をｵﾝします
>PR O_SW(0)
 1                          <-確認
>OFF_AND 1 0 0              <-OFF_AND結果が0なので出力は保持されます
>PR O_SW(0)
 1                          <-なるほどｵﾝのまま
>OFF_AND 1 1 0              <-OFF_AND結果が1になると出力0はｵﾌされます
>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】　機能：Ｉ／Ｏ　種別：コマンド　サポート：Ｐ

■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】　　機能：Ｉ／Ｏ　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　ON A1[,A2,A3]
OFF A1[,A2,A3]
A1,A2,A3：ポート番号
0≦A1,A2,A3≦255　Ｉ／Ｏ
-1≧A1,A2,A3≧-128 メモリー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。尚、ＯＦＦはオープンコレクタ出力がＯＦＦ状態でレベルはＨＩＧＨとなり、ＯＮはオープンコレクタ出力がＯＮ状態でレベルはＬＯＷとなります。以上ＯＮ，ＯＦＦは初期出荷状態では負論理(ＯＮにてシンクイン)と設定されていますがこれはＳＥＴＩＯコマンドによって変更する事ができます。(ＭＩＯ－８１６＃１～＃５迄のみ)

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

【ON_AND】　機能：Ｉ／Ｏ　　種別：コマンド　　サポート：Ｐ

■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なので当然出力はｵﾝしたままです
>PR O_SW(0)
 1
>ON_AND 1 0 0                  <-ON_ANDは結果が0の時には出力は保持されます
>PR O_SW(0)
 1                             <-OFF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ります
>PR O_SW(0)
 0
```

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

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

```
 0 0                              <--SW(17)=0 SW(18)=0
 0 1                              <--SW(19)=0 SW(20)=1
```

Ａ１、Ａ２は定数や変数を、ｎにはメモリーＩ／Ｏを用いることができます。

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

この他に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          <-とすればＳＷ(１６)がＯＮ、ＳＷ(１７)がＯＦＦ状態の時
                                   出力８がＯＮされます。
IF SW(16)=1 AND SW(17)=0 THEN ON 8
```

【ＯＮ＿ＯＲ】　機能：Ｉ／Ｏ　種別：コマンド　サポート：Ｐ

■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のどちらかかが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。

【ＯＰ】　機能：演算　種別：コマンド　サポート：Ｚ

■書　式　OP　文字列

■解　説　Ｚ版では、２byte整数インタプリタとして設定されている為4byteの長整数などの演算は不可能ですが、スタック演算を付加することにより次の演算を可能としています。スタック演算とはフォースなどが採用している演算の記述方法です。スタック領域としてＡＲ()が使用されますので注意して使用下さい。スタックされたデータへの演算命令は文字列で表現されます。文字は次の５種類で連続して記述できます。

／	除算	３２ｂｉｔ／１６ｂｉｔ －> 商１６ｂｉｔ 余り１６ｂｉｔ
＊	乗算	１６ｂｉｔ＊１６ｂｉｔ －> ３２ｂｉｔ
＋	加算	３２ｂｉｔ＋３２ｂｉｔ －> ３２ｂｉｔ
－	差算	３２ｂｉｔ－３２ｂｉｔ －> ３２ｂｉｔ
Ｒ	平方根	３２ｂｉｔ －> １６ｂｉｔ

通常の式からスタック演算への展開例

```
A=SQR(B*B+C*C)   ------>
```

```
100 PUSHI B,B
110 OP *
120 PUSHI C,C
130 OP *+R
140 A=PLI(0)
```

■Ｚ版で+/-32767を越える数値の計算■

スタック演算を利用するとＭＰＣ－８１６Ｚ版でも大きな数の計算を扱うことができます。たとえば、ＲＳ－２３２Ｃより次のフォーマットで８桁の数値を読みとるには次の方法によります。

±□□□□□. □□□ＣＲ／ＬＦ

```
S0=GET#(0)                      <-符号をＳ０に
C1=GET#(0)-48                   <-最初の一文字を１０進数
GETN# A1                        <-小数点までの４桁読
D1=GET#(0)                      <-小数点読み捨て
GETN#B1                         <-小数部を入力
SKIP# 10                        <-ＬＦまで読み捨てる
IF S0<>&H002D THEN *A1          <-Ｓ０がーで無ければＡ１
C1=-1*C1
A1=-1*A1                        <-負の場合の処理
B1=-1*B1
*A1
PUSHI 5000,C1,2000
OP **                           <-Ｃ１に10000000を乗ずる
PRINT LNG(0)                    <-ダミー表示
PUSHI B1,A1,1000                <-Ａ１を1000倍しＢ１を加
OP *++
PRINT LNG(0)                    <-スタックの中に８桁整数
```

また、スタックと変数や定数をインタフェースするコマンドは次の通りです。

【ＰＵＳＨＩ】A1[,A2,A3]

変数もしくは定数の値をスタックします。変数の内容はスタックオンする時に符号拡張し４byte整数に変換されます。

```
PUSHI １ 2 3
```

１，２，３の３つの値がスタックにセーブされます。

【ＰＵＳＨＤ】A1,A2

変数もしくは定数の値をスタックします。２つの値をそれぞれ上位下位として扱います。

```
PUSHD &HFFFF &HFFFE
```

4byte整数として－２がセーブされます。

【ＰＬＩ】　PLI(0)

スタックから下位２byteを取り出します。上位バイトは切り捨てられていますが次のＰＬＤ(０)によって値を得る事ができます。

【ＰＬＤ】　PLD(0)

上位２byteの値を得ます。この値は直前に実行されたＰＬＩ(０)により切り捨てられた上位の値です。

【ＬＮＧ】　LNG(0)

４byte整数としてデータをモニタします。表示できる数値の上限は絶対値で３２７６７０００００までです。この関数はデータを見るだけでスタックを変化させません。ＰＲＩＮＴと組み合わせて演算のデバッグに使用します。

```
100 PUSHI 10000
110 PUSHI -30000
120 OP *
125 PRINT LNG(0) (-3×10^8)
130 PUSHI 15000
140 OP /
150 A=PLI(0)
160 B=PLI(0)
```

（Ａには剰余、Ｂには商が入る）

【ＯＵＴ】　　機能：Ｉ／Ｏ　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　OUT n,m

n：セットする値

0≦n≦255　出力データ

m：ポートバンク

0≦m≦31　Ｉ／Ｏ

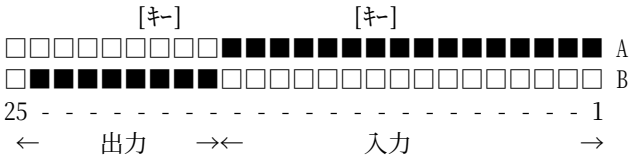
-16≦m≦-1　メモリーＩ／Ｏ

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

ＯＵＴコマンド使用例

```
OUT 255,0                       <-ﾊﾞﾝｸ0が全部ON
OUT 0,0                         <-ﾊﾞﾝｸ0が全部OFF
OUT &HF,1                       <-ﾊﾞﾝｸ1の下位4ﾋﾞｯﾄがON
OUT &HF0,1                      <-ﾊﾞﾝｸ1の上位4ﾋﾞｯﾄがON

10 FOR I=0 TO 255
20  OUT I,0                     <-ﾊﾞﾝｸ0にﾃﾞｰﾀIを出力
30  TIME 10
40 NEXT I
```

◇ＭＰＣ－８１６、ＭＩＦ－８１６、ＭＩＯ－８１６　Ｉ／Ｏコネクターピン配



◇ＭＰＣ－８１６シリーズ　バンク／ポート表

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

◇メモリーＩ／Ｏ　バンク／ポート表

-1	-25	-49	-73	-97	-121
-2	-26	-50	-74	-98	-122
-3	-27	-51	-75	-99	-123
-1 -4	-4 -28	-7 -52	-10 -76	-13 -100	-16 -124
-5	-29	-53	-77	-101	-125
-6	-30	-54	-78	-102	-126
-7	-31	-55	-79	-103	-127
-8	-32	-56	-80	-104	-128

-9	-33	-57	-81	-105
-10	-34	-58	-82	-106
-11	-35	-59	-83	-107
-2 -12	-5 -36	-8 -60	-11 -84	-14 -108
-13	-37	-61	-85	-109
-14	-38	-62	-86	-110
-15	-39	-63	-87	-111
-16	-40	-64	-88	-112

-17	-41	-65	-89	-113
-18	-42	-66	-90	-114
-19	-43	-67	-91	-115
-3 -20	-6 -44	-9 -68	-12 -92	-15 -116
-21	-45	-69	-93	-117
-22	-46	-70	-94	-118
-23	-47	-71	-95	-119
-24	-48	-72	-96	-120

■市販Ｄ／Ａボードの使用例

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

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

■市販Ａ／Ｄボードの使用例

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

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

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

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

【ＯＵＴ＿ＣＮＴ】　機能：ソフトカウンタ　　種別：コマンド　　サポート：Ｐ

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

```
90 'CONT CNTL
100 SETVAR C0,C9,0
110 C1=4
120 C3=5
130 C5=6
140 ON 0,1,2
150 OUT_CNT
```

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

【ＯＵＴ＿ＣＳＥＴ】機能：ソフトカウンタ　　種別：コマンド　　サポート：Ｐ

■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(例 in8,in9のみ監視は 0011b->3)
　　rev：１にしたビットを論理反転して検出

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

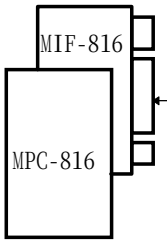
```
OUT_CSET 100 5 4
100：停止後１秒間カウンタを監視する
  5：in8,in10のみ監視
  4：n 10はB接点検出となる
```

【ＯＶＲＵＮ】　　機能：パルス　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　　OVRUN n
　　Ｐ版　n=&H■■□□ 入力24から31の入力パターン
　　■■：論理パターン

□□：マスクパターン
Z版　0≦n≦&HFF　入力24から31のビットパターン（ＯＮだけ有効）

■解　説



←― IN 24～31　オーバーランスイッチ

ビット	B7	B6	B5	B4	B3	B2	B1	B0
ﾎﾟｰﾄ番号	31	30	29	28	27	26	25	24

J=(IN(3) XOR ■■) AND □□
Jの値が0であれば停止しません。Jの値が0でなければ非常停止となります。論理パターンは各ビットを１（オン）で有効とするか０（オフ）で有効とするかを規定します。

```
OVRUN &H0FF                    <--ﾎﾟｰﾄ24から31のいずれかの入力ﾎﾟｰﾄがｵﾝになれば非常停止
OVRUN &HFFFF                   <--ﾎﾟｰﾄ24から31のいずれかの入力ﾎﾟｰﾄがｵﾌになれば非常停止
```

◆ＲＥＶによるオーバーラン状態の違い
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、ＲＥＶにより次の様な違いがあります。
Ｐ版　パルス発生コマンドでハングアップ。復帰はプログラムの再実行による。
Ｚ版　パルス発生コマンドから抜け次のステップに進む。つまりプログラムの監視によりオバーラン状態からの復帰が可能。

■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（Ｐ版）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(Ｐ版)
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)

【Ｚ版について】
このコマンドで開かれたビットがＯＮになるとパルス出力を停止します。非常停止は各パルス出力コマンドにたいして有効です。例えばポート３１と３０をオーバーラン入力とすると

```
OVRUN &HC0
```

とします。
＊ＭＯＤＥ　１ではＯＶＲＵＮを０以外に設定すると最大速度は２５kppsとなります。
＊ＡＣＣＥＬはＯＶＲＵＮ設定後再度実行して下さい。
＊ＯＶＲＵＮは指定したビットだけを監視しています。それ以外のポートはオンでもオフでも影響ありません。

【Ｐ】　機能：パルス　種別：配列　サポート：Ｐ・Ｚ

■書　式　P(n)

n：点番号
　Ｐ版　0≦n≦300
　Ｚ版　0≦n≦255

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

```
PRINT P(n)
111 222
```

注)４次元の座標表示は、

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

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

【ＰＡＬＥＴ】　　機能：パルス　　種別：コマンド　　サポート：Ｐ・Ｚ

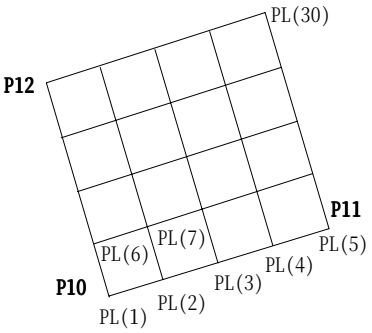
■書　式　PALET i,j,k

i,j,k：ティーチングされた点の番号

Ｐ版　1≦i,j,k≦300

Ｚ版　1≦i,j,k≦200

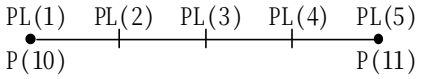
■解　説　この命令はセットで使用されます。ＰＡＬＥＴでパレットを決定する点を指定し、ＭＴＲＸにてパレットの縦、横の数を定義しています。(次の図参照)




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

パレット宣言によってＰＬ(１)～ＰＬ(３０)の点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。(Ｕ及びＺについてはＰ(ｉ)のＵＺ成分となる)各座標の値を必要とする場合は、ＰＬＸ(ｎ)，ＰＬＹ(ｎ)を使用します。それぞれＸ成分、Ｙ成分となります。１次元のパレットではｉとｋを同じ点番号とします。

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



【Q】ＰＡＬＥＴ、ＭＴＲＸコマンドの後にＨＯＭＥを実行するとパレタイズの座標値はクリアーされるか。

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

```
10      PG 2
20      MODE 5
30      ACCEL 30000
40      FEED 0
50      SETP 1,0,0
60      SETP 2,1000,1000
70      SETP 3,1000,1000
80      MOVE P(3)
90      PRINT P(0)
100     PALET 1,2,3
110     MTRX 11,1
120     GOSUB *HOME
130     'X=
140     PRINT STR(-1),X(0)
150     'Y=
160     PRINT STR(-1),Y(0)
170     '
180     FOR I=1 TO 11
190      MOVE PL(I)
200      PRINT PL(I)
210     NEXT I
220     END
230     *HOME
240     HOME 0
250     RETURN
>RUN
 1000 1000 0 0
```

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

FORK 4,10
>
 1000 1000 0 0
 X= 0
 Y= 0
 0 0 0 0
 100 100 0 0
 200 200 0 0
 300 300 0 0
 400 400 0 0
 500 500 0 0
 600 600 0 0
 700 700 0 0
 800 800 0 0
 900 900 0 0
 1000 1000 0 0
```

【Q】ＰＡＬＥＴコマンドは２つしか無く、ＰＡＬＥＴ　ＴＯ　ＰＡＬＥＴの移動で足りなくなる
【A】ＰＡＬＥＴ　ＴＯ　ＰＡＬＥＴなどの移動でＰＡＬＥＴを複数使う場合でも移動前に再宣言すると、いくつものＰＡＬＥＴに対応できます。時間的ロスも殆どありません。

```
10      PG 1
20      ACCEL 10000
30      FEED 0
40      SETPOS 0,0
50      '
60      SETP 1,0,0
70      SETP 2,1000,1000
80      SETP 3,1000,1000
90      SETP 11,2000,2000
100     SETP 12,4000,4000
110     SETP 13,4000,4000
120     '
130     *LOOP
140     FOR I=1 TO 11
150      GOSUB *PALET1
160      MOVE PL(I)
170      'PALET 1
180      PRINT STR(-1),X(0),Y(0)
190      '
200      GOSUB *PALET2
210      MOVE PL(I)
220      'PALET 2
230     PRINT STR(-1),X(0),Y(0)
240     NEXT I
250     GOTO *LOOP
260     '************
270     *PALET1
280     PALET 1,2,3
290     MTRX 11,1
300     RETURN
310     '************
320     *PALET2
330     PALET 11,12,13
340     MTRX 11,1
350     RETURN
>RUN
 PALET 1 0 0
 PALET 2 2000 2000
 PALET 1 100 100
 PALET 2 2200 2200
 PALET 1 200 200
 PALET 2 2400 2400
```

【ＰＡＬＥＴ１】　機能：パルス　種別：コマンド　サポート：Ｐ

PALET参照

【ＰＡＵＳＥ】　機能：タスク操作　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　PAUSE n[,m,1]

n,m,1：タスク番号

Ｐ版　1≦n,m,1≦11

Ｚ版　1≦n,m,1≦7

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

【ＰＧ】　機能：パルス　種別：コマンド　サポート：Ｐ

■書　式　PG n

n:ＭＰＧ－３０３ボードアドレス　1≦n≦3

-1を指定するとＭＩＦ－８１６のＪ５からパルス発生

■解　説　Ｐ版では、３枚のＭＰＧ－３０３を扱うことが出来ます。(３．２２ｅ以前では２枚までです)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　ショートピン１がショートされたものです。

ＴＡＳＫ４～７　ＭＰＧ＃２　ショートピン２がショートされたものです。

ＴＡＳＫ８～１１　ＭＰＧ＃３　ショートピン３がショートされたものです。

Ｔコマンド下では、ＴＡＢキーによりＭＰＧ－３０３を切り換える事ができます。ＰＧコマンドはタスク０に対してのみ有効なコマンドです。

```
MODE 5                          タスク０
PG 1                            ＰＧコマンドでＭＰＧを切り替えてＡＣＣＥＬ
ACCEL 5000                      の設定やＳＴＯＰコマンドを実行しています。
FEED 0
PG 2
ACCEL 5000
FORK 1,*PG1
FORK 4,*PG2
INPUT A
PG 1
STOP 1
PG 2
STOP 1
WAIT BSY(1)<>0
WAIT BSY(2)<>0
QUIT 4,1
END
*PG1                            タスク１
MOVE 10000,10000                このタスクのパルス命令はＭＰＧ　＃１で実行される。
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で実行される。
WAIT BSY(0)=1
TIME 10
```

```
MOVE 0,0
WAIT BSY(0)=1
TIME 10
GOTO *PG2
```

☆１～１１のタスクでＰＧコマンドを使うと、その効果は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が有効になってしまいます。
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ＳＷ１６～２３は原点センサー、ＳＷ２４～３１は停止入力ポート(ＯＶＲＵＮ)となります。ＳＴＯＰ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からﾊﾟﾙｽを出すﾓｰﾄﾞ
    ACCEL 3000              "ACCELは必須、MODEは不要
    FEED 0
    FEDZ 0
    OVRUN &HFF              "SW(24)～(31)のどれかがONで停止
    'OVRUN &H01             "SW(24)がONで停止。使わないﾎﾟｰﾄは一般入力OK
    'OVRUN &HFFFF           "SW(24)～(31)のどれかがOFFで停止
    'OVRUN &HFF01           "SW(24)がOFFで停止。使わないﾎﾟｰﾄは一般入力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 ｽﾋﾟｰﾄﾞ
    HOME &H0F 1000 1000     "停止ﾊﾟﾀｰﾝ SW(16)~(19)までON、退避移動CW 1000
    '                       "使わないﾎﾟｰﾄは一般入力OK
    'HOME COMP
    PRINT STR(-1)
    TIME 5
    SHMZ 1 4 400            "UCW ZCW  ｽﾋﾟｰﾄﾞ
    HOMZ &H0F 1000 1000     "停止ﾊﾟﾀｰﾝ SW(17)~(23)までON、退避移動CW 1000
    '                       "使わないﾎﾟｰﾄは一般入力OK
    'HOMZ COMP
    PRINT STR(-1)
    TIME 5
    GOTO *__RTN
```

```
        '===========
    *__RTN
        RETURN
```

【ＰＧＳ】　　機能：パルス　　種別：関数　　サポート：Ｐ

■書　式　PGS(n)

0≦n≦3

■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。

ｎ＝０　　タスクに対応するＭＰＧの状態を得る

ｎ＝１，２，３　　ＭＰＧ １～３を指定する

【ＰＬ】　　機能：パルス　　種別：関数　　サポート：Ｐ・Ｚ

■書　式　PL (n)

PL1 (n)

n：パレット上の点番号

1≦n≦32767

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

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

【ＰＬ１】　　機能：パルス　　種別：関数　　サポート：Ｐ

PL参照

【ＰＬ１Ｘ】　　機能：パルス　　種別：関数　　サポート：Ｐ

PLX参照

【ＰＬ１Ｙ】　　機能：パルス　　種別：関数　　サポート：Ｐ

PLX参照

【ＰＬＤ】　　機能：演算　　種別：関数　　サポート：Ｚ

OP参照

【ＰＬＩ】　　機能：演算　　種別：関数　　サポート：Ｚ

OP参照

【ＰＬＳ】　　機能：パルス　　種別：コマンド　　サポート：Ｐ

■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、停止の場合は 'Q 'を押し継続の場合はその他のキーを押します。

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

【ＰＬＳ＿ＭＩＦ】　機能：パルス　　種別：コマンド　　サポート：Ｐ

■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。

【ＰＬＸ】　機能：パルス　　種別：関数　　サポート：Ｐ・Ｚ

■書　式　　PLX (n)
PLY (n)
PL1X (n)
PL1Y (n)
　　n：ﾊﾟﾚｯﾄ上の点番号
　　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
```

【ＰＬＹ】　機能：パルス　　種別：関数　　サポート：Ｐ・Ｚ

PLX参照

【ＰＲ】　機能：デバッグ　　種別：コマンド　　サポート：Ｐ

■書　式　　PR A1[,A2,A3]
　　A1,A2,A3：定数，変数

■解　説　　ＰＲＩＮＴの省略形です。

【ＰＲＣ】　　機能：ＬＣＤ　　種別：コマンド　　サポート：Ｐ

■書　式　　PRC n[,m,l]
　　n,m,l：キャラクタ(アスキーコード)

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

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

【ＰＲＤ】　　機能：ＬＣＤ　　種別：コマンド　　サポート：Ｐ

■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。
　　ＬＣＤ関係のコマンド：LOC,PRC,PRD,PRS

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

【ＰＲＩＮＴ】　　機能：デバッグ　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　　PRINT A1[,A2,A3]　省略形　PR(Ｐ版のみ)
　　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(ｎ)	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

【ＰＲＩＮＴ＃】　　機能：ＲＳ－２３２Ｃ　　種別：コマンド　　サポート：Ｐ・Ｚ

■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)"[ｽﾍﾟｰｽ]"(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ＰＵＴＳ＃コマンドを使います。
◆ＰＲＩＮＴ＃、ＰＵＴＳ＃は使用前にＣＮＦＧ＃でＲＳ－２３２Ｃの初期化を行って下さい。

【ＰＲＳ】　機能：ＬＣＤ　種別：コマンド　サポート：Ｐ

■書　式　PRS n
n：表示文字数

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

```
LOC 1,2
'accel
PRS 5                          "1行2文字目から'accel'(5文字)を表示
```

【ＰＲＸ】　機能：デバッグ　種別：コマンド　サポート：Ｐ

■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
```

【ＰＵＬＳＥ】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　PULSE n[,D1,D2]
n：パルス発生数
D1：ＯＮ時間　単位100μsec
D2：ＯＦＦ時間　単位100μsec
-32767≦n≦32767
1≦D1,D2≦32767
（Ｐ版でも２byteの範囲です）

■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以下で使用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コマンドでは座標管理を行いません。

```
AXIS 1                         <-X軸指定
PULSE 1000                     <-ON 1msec,OFF 1msecで1000CWﾊﾟﾙｽ
PULSE -1000,100                <-ON 10msec,OFF 10msecで1000CCWﾊﾟﾙｽ
PULSE 2000,100,10              <-ON 10msec,OFF 1msecで2000CWﾊﾟﾙｽ
```

◆ＰＵＬＳＥコマンドパルスレート

	REV 2.2 (MIF から出力)	REV 3.2 (MPG-303P から出力)
PULSE 1000 10 10	499.052Hz	500.466Hz
9 9	553.81	555.425
8 8	622.073	623.943
7 7	709.52	711.746
6 6	825.58	828.307
5 5	987.04	990.523
4 4	1.227K	1.232K
3 3	1.621K	1.628K
2 2	2.388K	2.401K
1 1	4.533K	4.573K

【ＰＵＬＳＥ裏機能について(ｎ，ｍを負の値で指定した場合)　ＭＰＣ－８１６Ｘ　Ｚ版のみ】

■書　式　PULSE &HB(pat)(adr)-n-m

B (1bit)　B=0 cw,b=1 ccw
pat (7bit)　マスクパタン
adr (8bit)　入力ポートアドレス
n (0～32766) onデューティ時間　：(n-1)×18+65 クロック
m (0～32766) offデューティ時間　：(m-1)×18+64 クロック

■解　説　ＰＵＬＳＥコマンドでｎ，ｍを負の値に設定すると入力検出パルス発生となります。例えば次の様にコマンドを与えると

```
AXIS 1
PULSE &H8250 -100 -100
```

この場合では２．５kppsの５０％デューティのパルスで５０ＨつまりＭＩＦ－８１６の入力０～６までの間を検出し２とマスクをとるということになります。このため、入力１７がＯＮで停止となります。
パルスの計算は次の式によります。

T=129+(n-1)*18+(m-1)*18
Hz=9216000/T

この場合はm，ｎとも１００ですからＴ＝３６９３で周波数は約２．５kppsとなります。

```
AXIS 1
PULSE &H0248 -10 -190
```

この場合、ＣＷ出力となりパルスレートは同一です。

ＯＮ時間は　(9×18+65)/9216000=24.6μsec
ＯＦＦ時間は (189×18+64)/9216000=376.1μsec

となりデューティーは６．５％となります。
注)ＭＰＣ－８１６ＫではＣＰＵ，クロックともＭＰＣ－８１６Ｘと異なり上記の計算は成立しませんが、内部補正により準じた結果となります。

【Q】2kppsで加減速なしで５０％デューティーのパルスを出力したい。
【A】ＰＵＬＳＥコマンドでは2kpps付近での精度が悪いのでマシン語でプログラムした方がよい。次記マシン語サンプル。マシン語プログラムの組み込みは、システムローダによるバージョンアップと同じ手順です。Ｐ版であれば、Ｐ８１６Ｋ．８１６のマップファイルに組み込むヘキサファイルを追加します。Ｐ版、Ｚ版ではモジュール位置が異なります。組み込む版に合わせリンク時にアドレスを変更して下さい。

◆リンク例

Ｐ版　XL80 -y -n PULSE.HEX -x -p6E00 -dF930 PULSE
Ｚ版　XL80 -y -n PULSE.HEX -x -p5800 -dF930 PULSE

◆マシン語サンプルプログラム(±32767ﾊﾟﾙｽ、XCW/XCCWのみ)

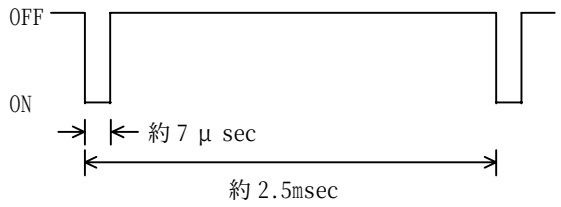
```
        .Z80
; 2kpps    rate=90 2.001khz
;          rate=91 1.980
;          rate=89 2.024
;
;   Fhz=921600/(50*rate+104)
;   CPU HD64180
;
; mpc-816x z 版
; 10 CALL &H5800 rate dummy  'パルスレート設定
; 20 CALL &H5804 acnt dummy  'パルス数設定  正:XCW;負:XCCW
;
; DSEG must be locate on &HF930
; CSEG must be locate on &H5800
;
; mpc-816x p 版
; 10 CALL &H6E00 rate dummy
; 20 CALL &H6E04 acnt dummy
;
; DSEG must be locate on &HF930
; CSEG must be locate on &H6E00
;
        DSEG
RATE:   DS    2
AXIS:   DS    1
;
        CSEG
        JP      SETC
        NOP
        JP      PULSE
;
SETC:   LD      (RATE),BC
        RET
PULSE:
        LD      A,1
        LD      (AXIS),A
        BIT     7,B
        JR      Z,$POS
        LD      HL,0
        AND     A
        SBC     HL,BC
        LD      B,H
        LD      C,L
        LD     A,2
        LD      (AXIS),A
$POS:
        LD      HL,(RATE)
        DI
$PULSE:
        LD      A,(AXIS)
        OUT     (40H),A
        CALL    TIME
        XOR     A
        OUT     (40H),A
        CALL    TIME
        DEC     BC
        LD      A,B
        OR      C
        JR     NZ,$PULSE
        EI
        RET
TIME:
        LD      D,H
        LD      E,L
$TIME:
        DEC     DE
        LD      A,D
        OR      E
        RET     Z
        JR      $TIME
;
        END
```

■単発パルス発生時のＰＵＬＳＥコマンドとＲＭＯＶコマンドの実行スピード
次のプログラムの文番号７０、８０を変更して実行した時のパルス間隔の実測値です。(次の絵のスケールはいい加減です)ＲＭＯＶは前後に内部的な処理があり、この様な１パルスずつのＬＯＯＰではＰＵＬＳＥより遅くなります。(計測MPC-816X REV3.22gをCHGREV 22,'93/12/03)

```
5       CHGREV 22                (*MPC-816KではCHGREVは不要)
10      MODE 1
20      ACCEL 30000
30      AXIS 1
40      FEED 0
50      FORK 1,*TASK1
60      *LOOP
70      'RMOV 1
80      PULSE 1,1,1
90      GOTO *LOOP
100     *TASK1
110     IF SW(1)=1 THEN *DUMMY
120     IF SW(2)=1 THEN *DUMMY
130     IF SW(0)=0 THEN *DUMMY
140     *DUMMY
150     ON 0
160     TIME 10
170     OFF 0
180     TIME 10
190     GOTO *TASK1
```

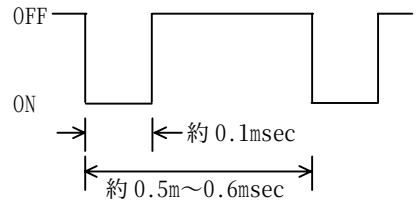
■ＲＭＯＶの場合

```
60    *LOOP
70     RMOV 1,1,1
80     'PULSE 1,1,1
90     GOTO *LOOP
```



■ＰＵＬＳＥの場合

```
60    *LOOP
70     'RMOV 1,1,1
80     PULSE 1,1,1
90     GOTO *LOOP
```



【ＰＵＳＨＤ】　機能：演算　種別：コマンド　サポート：Ｚ

OP参照

【ＰＵＳＨＩ】　機能：演算　種別：コマンド　サポート：Ｚ

OP参照

【ＰＵＴ】　機能：ＲＳ－２３２Ｃ　種別：コマンド　サポート：Ｐ

■書　式　PUT A1[,A2,A3]

A1,A2,A3：変数／定数

0≦A1,A2,A3≦&H7F

■解　説　Ａ１，Ａ２，Ａ３の値をアスキーコードとしてＣＨ０に出力

```
PUT &H20                 <-[ｽﾍﾟｰｽ]を出力します。
PUT &H41                 <-[A]と出力
PUT 65,66,67             <-[A][B][C]と出力
```

【ＰＵＴ＃】　　機能：ＲＳ－２３２Ｃ　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　PUT# A1[,A2,A3]

A1,A2,A3：変数／定数

0≦A1,A2,A3≦&H7F

■解　説　Ａ１，Ａ２，Ａ３の値をアスキーコードとしてＣＨ１に出力

```
PUT# &H20                       <-[ｽﾍﾟｰｽ]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通信ﾊﾞｯﾌｧのｷｬﾗｸﾀｰ数
70 IF R1<>0 THEN 40
80 FOR I1=1 TO I
90 PRINT AR(I1)
100 NEXT I1
RUN
65
66
67
68
69
13
```

（＊このプログラムはコネクタ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キャラクターずつ出力します。

```
CNFG# 4,0,2                     ノンパリ、データ８ビット、９６００ボー
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＃】　　機能：ＲＳ－２３２Ｃ　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　PUTS# A1[,A2,A3]

A1,A2,A3：変数，定数

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

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

【ＱＵＩＴ】　　機能：タスク操作　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　QUIT A1[,A2,A3]

A1,A2,A3：タスクナンバー

Ｐ版　1≦A1,A2,A3≦11

Ｚ版　1≦A1,A2,A3≦7

■解　説　　ＦＯＲＫされたプログラムの停止です。ＦＯＲＫされたプログラムが、ＥＮＤで終了していれば不要です。
※ＭＰＧ－３０３を使用中のタスクには直接ＱＵＩＴを実施しないで下さい。必ずＳＴＯＰ　４やＢＳＹ(ｎ)を併用して下さい。

【ＲＥＧ】　　機能：ＭＰＧ－３０１　　種別：コマンド　　サポート：Ｐ

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

■解　説　　ＭＰＧ－３０１に搭載されているパルス発生ＩＣ「Ｘ３２０２」のレジスタのデータを読込みます。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。
ＭＰＧ－３０１のコマンド／関数：CMND,REG(),REG3(),ST_REG
参照：「ＭＰＧ－３０１ 詳細マニュアル」

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

【ＲＥＧ３】　　機能：ＭＰＧ－３０１　　種別：コマンド　　サポート：Ｐ

■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。
ＭＰＧ－３０１のコマンド／関数：CMND,REG(),REG3(),ST_REG
参照：「ＭＰＧ－３０１ 詳細マニュアル」

```
C=REG3(&H21)                   "MPG-301 #1 ｶｳﾝﾀA読込み
```

【ＲＥＭ】　　機能：編集　　種別：コマンド　　サポート：Ｐ・Ｚ

■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。尚、コメント文はＳＴＲ()関数を文字列定数として使用できます。

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

【ＲＥＮＵＭ】　　機能：編集　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　　RENUM [n,o,s]
n：新文番号

o：旧文番号
s：ステップ量
0≦n,o≦32766
1≦s≦1000

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

【ＲＥＴＵＲＮ】　機能：制御文　種別：コマンド　サポート：Ｐ・Ｚ

■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。

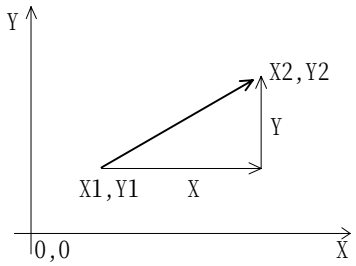
```
90 OFF 1
100 GOSUB 1000
110 ON 1
120
    |
1000 'AYA
1010 WAIT SW(1)=1
1020 RETURN
```

この様に、ＲＥＴＵＲＮはＧＯＳＵＢの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。

【RMOV】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　RMOV X,Y

■解　説　モード５，６ともコマンド上の扱いは全く同等です。ＸＹ軸に対してX，Yパルス出力します。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。(直線補間)この時の現在位置をＸ１，Ｙ１とするとパルス発生後の現在位置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。

【Z版について】
相対座標移動(現在位置より何パルス出力か)パルス発生です。スピードの変更はＦＥＥＤコマンドによって行います。パルス発生にはモード１～モード４までありそれぞれに特性の異なった動作をします。

ＭＯＤＥ１
■書　式　RMOV X,Y[,U]　-32767≦X,Y,U≦32767
■解　説　X,Y,U：各軸の発生パルス数
ＲＭＯＶコマンドは引数によって指定された数だけパルス発生します。(同時３軸)このパルス発生によって移動した数量だけ現在位置データは変更されます。Ｕを略すると０と扱われます。

ＭＯＤＥ２、ＭＯＤＥ３

■書　式　　RMOV X,Y,J　　-32767≦X,Y≦32767

■解　説　　X,Y：各軸の発生パルス数　Ｊ：停止条件

ＭＯＤＥ２、ＭＯＤＥ３はX，Ｙの２軸移動です。３番目の引数は停止条件で移動中に入力条件によって停止する事ができます。停止条件を与えない場合はＪ＝０。

ＭＯＤＥ４

■書　式　　RMOV X,J　　-32767≦X≦32767

■解　説　　X：発生パルス値　　Ｊ：停止条件

１軸のパルス発生コマンドです。出力軸はX～Ｚを選択できます。ＭＯＤＥ４はＡＸＩＳコマンドで出力軸を選択します。

ＡＸＩＳ　１にて　パルス出力ポートは　X軸
ＡＸＩＳ　２にて　パルス出力ポートは　Ｙ軸
ＡＸＩＳ　３にて　パルス出力ポートは　Ｕ軸
ＡＸＩＳ　４にて　パルス出力ポートは　Ｚ軸

となります。停止条件がＪが不要の場合はＪを０として下さい。

※停止条件は、ＳＷ(２４)～ＳＷ(３１)までのビットパタンと停止モードです。停止モードの選択は上位８bit、入力ビットパタンは下位８bitで設定します。

Ｊ＝＆Ｈ××△△
××＝０　減速停止
××≠０　急停止

△△の表

ビット	Ｂ７	Ｂ６	Ｂ５	Ｂ４	Ｂ３	Ｂ２	Ｂ１	Ｂ０
入力ポート	３１	３０	２９	２８	２７	２６	２５	２４

停止条件はビットによって指定されたポートの論理和となります。例えば、Ｊ＝３とします。この場合ポート２４、ポート２５のどちらかがＯＮとなればパルスの出力は停止されます。なお、停止条件Ｊの使用方法はＲＭＯＶ、ＭＯＶＺ、ＲＭＶＺのいずれとも共通です。ＭＯＶＥコマンドは絶対座標位置に移動するのに対してＲＭＯＶは相対移動をします。

【Q】Y軸だけを移動させたいときに、次の様なＲＭＯＶをしたところ、Ｙ軸のスピードが低下した。

```
RMOV X(0),n  (nは移動量)
```

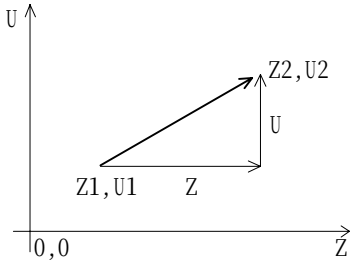
【A】この記述をするとX軸には現在点の座標値、Ｙにはｎが入りまから、この命令ではX軸の現在の座標値が０で無い限りX軸のパルスも出力されます。もしXの座標がＹより大きな数値の場合はX軸が主軸になるのでＹ軸のパルスレートは遅くなります。現在Ｘ＝１００００とするとＲＭＯＶ　Ｘ(０)，１００ではＲＭＯＶ　１００００，１００になり、主軸はX軸になります。Ｙ軸のみの相対移動はＲＭＯＶ　０，ｎとするのが正当な記述です。(たぶんＭＯＶＥと勘違いしているのではないかとおもいますが．．．)絶対移動命令ＭＯＶＥでの単軸移動にはＸ(０)などの現在点取得関数を用います。

```
MOVE X(0),1000      <-Xは動かず、Yは座標値1000に移動。

A=Y(0)+1000         <-Xは動かず、Yは現在位置から+1000移動。ARMOV 0,1000と同じ
MOVE X(0)
```

【ＲＭＶＺ】　　機能：パルス　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　　Ｐ版　RMVZ Z,U
Z：Ｚ軸移動量
U：Ｕ軸移動量
Ｚ版　RMVZ Z,J
Z：パルス出力値
-32767≦Z≦32767
Ｊ：停止条件

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



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。

【Ｚ版について】

ＲＭＶＺはモード１～３に対して有効で、モード４では無視されます。ＲＭＶＺはＺ軸に対する相対移動命令で、ＲＭＯＶと同様指定された数だけパルス出力されます。停止条件ＪはＭＯＶＥと同じです。スピードの変更はＦＥＤＺを用います。

【ＲＳ】　　機能：ＲＳ－２３２Ｃ　　種別：関数　　サポート：Ｐ・Ｚ

■書　式　RS(n)

n：ＲＳ－２３２Ｃチャンネル番号　0または1

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

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

【ＲＳＶ】　　機能：タスク操作　　種別：関数　　サポート：Ｐ

■書　式　RSV(n)

-1≧n≧-128 （メモリＩ／Ｏ）

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

```
FORK 1,*MPG1
FORK 2,*MPG2
   (略)

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

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

【ＲＵＮ】　　機能：制御文　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　RUN n

n：文番号(存在する文番号)
1≦n≦32766
またはラベル

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

【ＳＥＴ】　　機能：パルス　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　SET n,l,J

n：インチング量バンク指定
0≦n≦3
l：Ｘ軸，Ｙ軸，Ｚ軸のインチング量
J：Ｕ軸のＪＯＧ量
-32767≦l,J≦32767

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

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

【ＳＥＴＩＯ】　　機能：Ｉ／Ｏ　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　SETIO &HXX

XX：ヘキサ

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

ＢＩＴ０　　ポートバンク０
ＢＩＴ１　　〃　　１
ＢＩＴ２　　〃　　２
・　　・
・　　・
・　　・
ＢＩＴ６　　〃　　６

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

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

【ＳＥＴＰ】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　SETP n,ｘ,ｙ

n：点番号

x,y：変数，定数

■解　説　点ｎに座標ｘ，ｙを入力します。点ｎへのＸ，Ｙ座標値の設定

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

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

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

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

【ＳＥＴＰＯＳ】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　SETPOS X Y

X,Y：Ｘ軸，Ｙ軸座標

Ｐ版　-8388607≦X,Y≦8388607

Ｚ版　-32767≦X,Y≦32767

■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。尚、Ｚ版のＭＯＤＥ　４ではＸ軸のみ有効、またＸ軸の値はＡＸＩＳで選択された座標に設定されます。

【ＳＥＴＶＡＲ】　機能：演算　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　SETVAR V1,V2[,n]

V1,V2：変数

n：変数、定数

■解　説　ＭＰＣ－８１６の変数は次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にします。
SETVAR B0 B 0                   <-B行の変数を0にします。
SETVAR C2 C9                    <-C2,C3,C4,C5,C6,C7,C8,C9を0にします。

A=123
SETVAR D0 D A                   <-D行の変数を123にします。
```

【ＳＦＤＬ】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

■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ＳＦＤＬでは、ＦＥＤＺ　１５での加速パルス数を強制的に変更することができます。従って、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
```

ここでは、加速距離１パルスのため全く加速しません。従って著しく遅いスピードを得ることが出来ます。尚、加速パルス数については、ＡＣＣＥＬを参照下さい。

【ＳＦＴＬ】　機能：演算　種別：コマンド　サポート：Ｐ・Ｚ

■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)→

シフトレジスタとして使用することができます。参照　ＡＲ()

【ＳＦＴＲ】　機能：演算　種別：コマンド　サポート：Ｐ・Ｚ

■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(３)→

シフトレジスタとして使用することができます。

【ＳＨＭＺ】　機能：パルス　種別：コマンド　サポート：Ｐ

■書　式　SHMZ u,z,s

u:U軸原点復帰方向

u=1(CW),2(CCW),0(無し)

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

■解　説　　原点復帰(ＨＯＭＺ)のｚ，ｕ軸パルスの出力方向を定めます。

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

【ＳＨＭＺＵ】　　機能：パルス　　種別：コマンド　　サポート：Ｚ

■書　式　　SHMZU Z,U
　　Z,U：退避逆回転量
　　0≦Z,U≦32767

■解　説　　原点位置から再度原点復帰を実施する場合は、１度原点から退避する必要があります。ＳＨＭＺＵは、この時のＺ軸及びＵ軸の移動量を定めます。ＨＯＭＥもしくはＨＯＭＺを実行する時、この値が使用されます。尚、Ｚ版でのＨＯＭＥは、Ｘ，Ｙ，Ｕ３軸の原点復帰となっておりＵ軸の原点復帰方向ＣＣＷに固定されています。Ｐ版はＳＨＭＺ参照

【ＳＨＯＭ】　　機能：パルス　　種別：コマンド　　サポート：Ｐ・Ｚ

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

■解　説　　原点復帰(ＨＯＭＺ)のｘ，ｙ軸パルスの出力方向を定めます。

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

尚、ｓと原点復帰時のパルスレートの関係は次の通りです。

Ｐ版　モード５及びモード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。

Ｚ版　モード１～３の場合
　　F=9216×10e3/(408+(60×s+424)×n)
ｎの値はＳＨＲＤによって変更されます。デフォルトは２です。ｓを１０の時のパルスレートは約３．９Khzとなります。

Ｚ版　モード４の場合
　　F=9216×10e3/(973+120×s)

モード４ではＳＨＲＤは無効です。ｓが１０の時約４．２Khzとなります。

【ＳＨＲＤ】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　SHRD n
2≦n≦10

■解　説　ＨＯＭＥ命令では、原パルスを出力する毎に原点入力をモニタし、ＨＯＭＥコマンドで指定されたセンサパタンと比較します。この比較の回数を定めるのが、ＳＨＲＤです。通常(リセット後)は、４と指定されていますが、サーボドライバなどのＺ相(Ｃ相)検出ではパルス幅の狭いことや、エンコーダの振動などから検出が困難です。この為、ＳＨＲＤ　６～ＳＨＲＤ　８を指定し、検出を確実なものとします。Ｚ版でのｎのデフォルトは２です。　ｎの値と原点復帰スピードの関係はＳＨＯＭ参照

【ＳＫＩＰ#】　機能：ＲＳ－２３２Ｃ　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　SKIP# n
n：アスキーコード
0≦n≦&H7F

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

【ＳＫＰＳＰ#】　機能：ＲＳ－２３２Ｃ　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　SKPSP#

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

【ＳＬＯＷ】　機能：メンテナンス　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　SLOW n
n：文字出力タイマー(１msec単位)
0≦n≦15

■解　説　ｎは１文字出力するごとのタイマーでｎを５とすると５msec間隔でキャラクタを送出します。遅いターミナルに適応する場合に有効で、ｎの最大値は１５。使用するターミナルが、処理速度が遅く、ＸＯＮ／ＸＯＦＦ制御の無い場合に使用します。ＳＬＯＷ　１で１キャラクタ出力する毎に１msecのタイマーが置かれ、およそボーレート４８００での送信速度に近くなります。ＭＰＣＩＮＩＴコマンドの初期化時、ＦＴＭの立ち上がり時には０になります。ＳＬＯＷ設定を大きくするとＬＩＳＴ表示やＳＡＶＥに時間がかかります。

【ＳＰ】　機能：ファイル　種別：コマンド　サポート：Ｐ

■書　式　点データの保存(セーブ)
SP n

n：ファイルメモリーナンバー
0≦n≦3

■解　説　点データの保存。P版では点データを通常使用しないメモリエリアに保存することが出来ます。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。

```
10      FOR I=1 TO 10
20        J=I*100
30        K=I*1000
40       SETP I,J,K  <-ﾎﾟｲﾝﾄﾃﾞｰﾀを作成します。
50      NEXT I
>RUN
>PLS                <-確認してみます。
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               <-ﾎﾟｲﾝﾄﾃﾞｰﾀを保存します。
>NEWP               <-ﾒｲﾝﾒﾓﾘｰのﾎﾟｲﾝﾄﾃﾞｰﾀをｸﾘｱｰ
>PLS                <-確認
P(1): 0 0 0 0
P(2): 0 0 0 0
P(3): 0 0 0 0
P(4): 0 0 0 0
P(5): 0 0 0 0
P(6): 0 0 0 0
P(7): 0 0 0 0
P(8): 0 0 0 0
P(9): 0 0 0 0
P(10): 0 0 0 0
ok
>LP 1              <-保存したﾎﾟｲﾝﾄﾃﾞｰﾀを読込ます
>PLS                <-確認
P(1): 100 1000 0 0
P(2): 200 2000 0 0
P(3): 300 3000 0 0
P(4): 400 4000 0 0
P(5): 500 5000 0 0
P(6): 600 6000 0 0
P(7): 700 7000 0 0
P(8): 800 8000 0 0
P(9): 900 9000 0 0
P(10): 1000 10000 0 0
ok
>
```

消えている

もとに戻った

【ＳＱＲ】　機能：演算　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　SQR(n)
n：変数，定数
0≦n≦32767

■解　説　ｎの平方根を求めます。

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

ｎを負の数とすると平方根は負の数で出力されます。

【ＳＱＲＴ】　機能：演算　種別：コマンド　サポート：Ｐ

■書　式　SQRT n m v

n,m：変数,定数 >0

v：結果が入る変数 (v=root(n*n+m*m))

■解　説　n､mの平方根を求めます。

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

【ＳＴ＿ＲＥＧ】　機能：ＭＰＧ－３０１　種別：コマンド　サポート：Ｐ

■書　式　ST_REG reg data

reg：ＭＰＧアドレスとＸ３２０２のレジスタ／カウンタセレクトコード

&Haaxx

aa：ＭＰＧアドレス(省略時#1)

xx：セレクトコード

data：設定データ

■解　説　ＭＰＧ－３０１に搭載されているパルス発生ＩＣ「Ｘ３２０２」のレジスタにデータを書き込みます。

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

参照：「ＭＰＧ－３０１ 詳細マニュアル」

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

【ＳＴＯＰ】　機能：パルス　種別：コマンド　サポート：Ｐ

■書　式　STOP n

n：停止モード

n=1　減速停止

n=2　急停止

n=3　Ｉ／Ｏ指定

n=4　ＭＰＧ－３０３停止

n=5　ＭＰＧ－３０３停止解除

■解　説　◇ｎ＝１，２の場合

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

◇ｎ＝３の場合

STOP 3,P,O

-128≦P≦512 ポート番号(#DEFS定義の文字列は不可,変数は可)

O=0,1　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
110 MOVE 20000,20000
```

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

◇ＳＴＯＰコマンドは出力されたパルス数と座標値は整合されています。

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

◇ＳＴＯＰはＨＯＭＥ、ＰＵＬＳＥコマンドにも有効です。

■ＭＰＣ－８１６　Ｐ版のＳＴＯＰコマンドによるパルス発生の途中停止の例です。（基本型）

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

※１タスク０でスイッチを監視してタスク１とタスク４でのパルス発生を停止するプログラムです。ＳＴＯＰ　１　コマンドを実行してもすぐにパルスは停止しません。もしパルス発生中にタスクをＱＵＩＴしてしまうとＰＧは異常な状態となります。ＳＴＯＰを実行した後はＢＳＹ()でパルスの停止を確認してタスクをＱＵＩＴします。
※２タスク０でＳＴＯＰコマンドを実行すると、実行されたＰＧがパルス発生中であればプログラムはそのスッテプから出て次の命令を実行します(ハングアップしない)。もしタイミング悪く次のパルス発生コマンド実行中にそのタスクがＱＵＩＴされると減速停止されなくなるため、正常停止以外は次の命

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

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

```
10 STOP 4
20 PG 1                          <-PGｺﾏﾝﾄﾞは大丈夫だが
30 MODE 6                        <-STOP 4がかけられたままだとﾊﾝｸﾞｱｯﾌﾟする。
```

◆ＳＴＯＰ４、ＳＴＯＰ　５の使用例　その①

```
10      PG 1
20      FORK 1,*TASK1
30      *LOOP
40      INPUT A
50      STOP 4                   <-MPG-303停止
60      WAIT BSY(0)<>0           <-ﾊﾟﾙｽ発生終了の確認
70      PAUSE 1                  <-ﾀｽｸﾎﾟｰｽﾞ
80      '#1 PAUSE
90      PRINT STR(-1)
100     INPUT A
110     STOP 5
120     CONT 1
130     '#1 CONTINUE
140     PRINT STR(-1)
150     GOTO *LOOP
160     *TASK1
170     MOVE 0,0
180     MOVE 100000,0
190     GOTO *TASK1
>RUN
?
 #1 PAUSE
?
 #1 CONTINUE
?
```

◆ＳＴＯＰ４、ＳＴＯＰ　５の使用例　その②
パルス発生が連続しているタスクを停止させるにはＳＴＯＰ　４でパルスコマンドをサスペンドさせてからＳＴＯＰ　１またはＳＴＯＰ　２を実行します。次の場合、もしＳＴＯＰ　４をしないとＳＴＯＰ　２で停止した直後に次のパルスコマンドを実行してしまい、パルスが停止しません。

```
 PG 1
 FORK 1,*PULSE
 INPUT A
 STOP 4                          <-STOP 4でﾊﾟﾙｽ発生ｺﾏﾝﾄﾞをｻｽﾍﾟﾝﾄﾞします。
 STOP 2                          <-STOP 1またはSTOP 2でﾊﾟﾙｽ発生を停止します。
 WAIT BSY(1)<>0
 QUIT 1
 PRINT TASK(1)
 STOP 5                          <-STOP 4のｻｽﾍﾟﾝﾄﾞを解除します。忘れずに！
 END
*PULSE
 MOVE 0,0
 MOVE 100000,100000
 GOTO *PULSE
```

■ＳＴＯＰコマンド使用上の注意
ＳＴＯＰ、ＪＯＧ、Ｘ(０)、Ｙ(０)、ＢＳＹ()などのパルス発生や現在点の取得の関数はＭＰＧ－３０３とのやりとりを行います。次のプログラムの様にこれらのコマンドや関数を連続して記述すると、パ

ルスが停止しなくなるなど、正常に動作しないことがあります。

うまく動かない

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

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

対策

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

【ＳＴＰＺＵ】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　STPZU n,z,u

ｎ：点番号
ｚ：ｚ座標の値
ｕ：ｕ座標の値
Ｐ版　1≦n≦300
Ｚ版　1≦n≦255

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

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

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

【ＳＴＲ】　機能：文字列　種別：関数　サポート：Ｐ・Ｚ

■書　式　STR(n)

ｎ：コメント文の文番号
1≦n≦32766 または -1　(-1はＰ版のみ有効で直前のコメント文を指定)

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

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

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

```
"a=1000 b-2000CR"
1000 'a=
1010 'b=
```

```
1020 PUTS# STR(1000),1000
1030 PUTS# STR(1010),2000
1040 PUT# &H0D
```

次のサブルーチンば“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。（Ｐ版のみ）

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

【SW】　機能：Ｉ／Ｏ　種別：関数　サポート：Ｐ・Ｚ

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

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

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

【！SW】　機能：Ｉ／Ｏ　種別：関数　サポート：Ｐ・Ｚ

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

【T】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

TEACH参照

【ＴＡＩＬ】　機能：編集　種別：コマンド　サポート：Ｐ

■書　式　TAIL

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

【ＴＡＳＫ】　機能：タスク操作　種別：関数　サポート：Ｐ

■書　式　TASK(n)

n：タスク番号

-1≦n≦11

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

０：タスク実行中
４：タスク一時停止中　　（ＰＡＵＳＥ状態）
７：タスク未使用　　（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)                  <-自分はどのﾀｽｸ?
 0                               <-自分はﾀｽｸ0
PRINT TASK(0) TASK(1)            <-ﾀｽｸ0と1の状態は?
 0 7                             <-ﾀｽｸ0は実行中、ﾀｽｸ1は未使用
>
```

【ＴＥＡＣＨ】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　TEACH または、T

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

［0］ インチング量０を選択します。
［1］ インチング量１を選択します。
［2］ インチング量２を選択します。
［3］ インチング量３を選択します。

［X］ ＋Ｘ　インチング量だけ移動します。
［x］ －ｘ　インチング量だけ移動します。
［Y］ ＋Ｙ　インチング量だけ移動します。
［y］ －ｙ　インチング量だけ移動します。
［U］ ＋Ｕ　インチング量だけ移動します。
［u］ －ｕ　インチング量だけ移動します。
［Z］ ＋Ｚ　インチング量だけ移動します。
［z］ －ｚ　インチング量だけ移動します。

［O］ ポートＯＮ
［F］ ポートＯＦＦ

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

［Q］ ティーチング・モード終了

［ ］ スペース　ＡＸＩＳ切り換え
［ＴＡＢ］　タブ　ＰＧ１～３選択(ＭＰＧ－３０３使用の場合)

※インチング・・・キーを１回押すごとに一定量移動すること。

☆ティーチングモードにはターミナルソフトからしか入れません。
操作パネルのスイッチなどでポイントを設定したいときはＪＯＧコマンドや、ＳＥＴＰ、ＳＴＰＺＵコマンドを使いプログラムします。

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

```
TEACH       <-TEACH または T でﾃｨｰﾁﾓｰﾄﾞ

PGﾅﾝﾊﾞｰ                          JOG 量(一回に動くﾊﾟﾙｽ数)です。0,1,2,3 の
TABｷｰで切替      座標値          ｷｰで変わります。設定変更は SETｺﾏﾝﾄﾞです。
┌──┐          ┌──────┐      ┌──────┐
PG 1#1(X,Y,Z,U) 0 0 0 0 [XYZ,U] 400 400
          ↓  Xｷｰをｲｯﾊﾟﾂ
PG 1#1(X,Y,Z,U) 400 0 0 0 [XYZ,U] 400 400
          ↓  Xｷｰをもうｲｯﾊﾟﾂ
PG 1#1(X,Y,Z,U) 800 0 0 0 [XYZ,U] 400 400
          ↓  Xｷｰをさらにｲｯﾊﾟﾂ
PG 1#1(X,Y,Z,U) 1200 0 0 0 [XYZ,U] 400 400
          ↓  Zｷｰをｲｯﾊﾟﾂ
PG 1#1(X,Y,Z,U) 1200 0 400 0 [XYZ,U] 400 400
          ↓  Zｷｰをもうｲｯﾊﾟﾂ
PG 1#1(X,Y,Z,U) 1200 0 800 0 [XYZ,U] 400 400
          ↓  Zｷｰをさらにｲｯﾊﾟﾂ
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ﾊﾟﾂ
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ﾊﾟﾂ
PG 1#1(X,Y,Z,U) 1200 0 2000 0 [XYZ,U] 400 400
          ↓
P1  <-ここぞというところで Pｷｰを押しﾎﾟｲﾝﾄﾅﾝﾊﾞｰを入力(ここが P(1)になる)
          ↓
PG 1#1(X,Y,Z,U) 1200 0 2000 0 [XYZ,U] 400 400
          ↓  Xｷｰを 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  <-ここぞというところで Pｷｰを押しﾎﾟｲﾝﾄﾅﾝﾊﾞｰを入力(ここが P(2)になる)
          ↓
PG 1#1(X,Y,Z,U) 3200 0 3600 0 [XYZ,U] 400 400
          ↓
Q     <-ﾃｨｰﾁﾝｸﾞﾓｰﾄﾞの終了は Qｷｰ

PLS                              <-PLSｺﾏﾝﾄﾞでﾎﾟｲﾝﾄの一覧を表示します。
P(1): 1200 0 2000 0
P(2): 3200 0 3600 0
P(3): 0 0 0 0
P(4): 0 0 0 0
P(5): 0 0 0 0
P(6): 0 0 0 0
P(7): 0 0 0 0
P(8): 0 0 0 0
P(9): 0 0 0 0
P(10): 0 0 0 0
ok
```

■プログラム例

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

【ＴＨＥＮ】

IF参照

【ＴＩＭＥ】　　機能：タイマー　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　TIME n

n：タイマー値　　n=1で10msec

Ｐ版　0≦n≦8388607

Ｚ版　0≦n≦32767

■解　説　時間待ちのタイマーで、１０msec単位で設定

TIME 100　(100×10msec=1000msec=1秒)

【TMOUT】　　機能：タイマー　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　TMOUT n

n：タイマー値　　n=1で10msec

Ｐ版　0≦n≦8388607

Ｚ版　0≦n≦32767

■解　説　ＷＳ１()、ＷＳ０()のタイムアウト時間を設定します。TMOUTは、随時変更することができますが最後に実行された値が全てのＷＳ１()，ＷＳ０()関数に適応されます。

【ＴＯＦＦ】　　機能：デバック　　種別：コマンド　　サポート：Ｐ・Ｚ

TON参照

【ＴＯＮ】　　機能：デバック　　種別：コマンド　　サポート：Ｐ・Ｚ

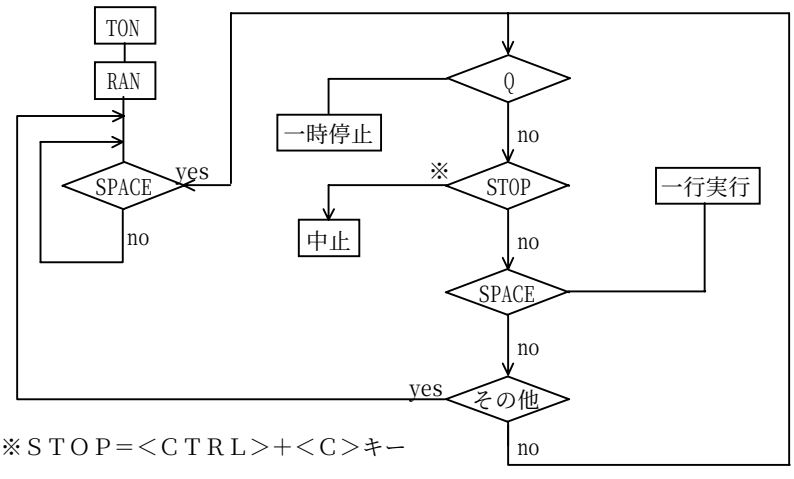
■書　式　TOFF

TON

■解　説　メイン・タスクのみに適用。ダイレクトコマンドで実行します。トレース・モードは＜ＳＰＡＣＥ＞キー入力によって一時停止します。一時停止中には次の操作ができます。

＜ＳＰＡＣＥ＞キー入力で、１ステップ実行
＜ＣＴＲＬ＞＋＜Ｃ＞キー入力で、中止
＜Ｑ＞キー入力で、コマンドモードへ復帰
その他のキー入力で、継続

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



※ＳＴＯＰ＝＜ＣＴＲＬ＞＋＜Ｃ＞キー

【ＴＳＴ＃】　　機能：ＲＳ－２３２Ｃ　種別：関数　　サポート：Ｐ・Ｚ

■書　式　TST#(n)

n：ダミー　0を設定して下さい。

■解　説　ＲＳ－２３２Ｃ　ＣＨ１の割り込み入力。バッファの先頭の文字がどの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。次の５種類です。

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

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

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

となります。これは、先頭の文字”A”が英字である為です。

【U】　　機能：パルス　　種別：配列　　サポート：Ｐ・Ｚ

■書　式　U(n)

n：点番号
Ｐ版　0≦n≦300
Ｚ版　0≦n≦255

■解　説　Ｕ座標を与えます。ｎ＝０の時は、現在位置となります。

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

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

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

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

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

【ＵＲＡＮＧ】　　機能：パルス　　種別：コマンド　　サポート：Ｐ・Ｚ

XRANG参照

【Ｖ＿ＳＷＡＰ】　　機能：メンテナンス　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　V_SWAP

■解　説　Ｐ／Ｚ版の切り替えをします。

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

```
 TNYFSC(R) Rev-2.50b [VER-Zmx1729]    <--Z版
 Copyright(C)by ACCEL CORP/BC-SOFT
 [255point MPC-816K MODE1~4]K8a9
>MPCINIT                              <--初期化
>ERASE
*
```

【ＶＥＲ】　　機能：メンテナンス　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　　VER

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

【ＶＬＩＳＴ】　　機能：デバック　　種別：コマンド　　サポート：Ｐ

■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
>
```

【ＶＰＬＳ】　　機能：パルス　　種別：コマンド　　サポート：Ｐ

WPLS参照

【ＷＡＩＴ】　　機能：制御文　　種別：コマンド　　サポート：Ｐ・Ｚ

■書　式　WAIT　～条件式～

■解　説　条件式が成立するまで停止します。

ＷＡＩＴ　ＳＷ(ｎ)＝０　：　ＳＷ(ｎ)がＯＦＦになるまで待つ

ＷＡＩＴ　ＳＷ(ｎ)＝１　：　ＳＷ(ｎ)がＯＮになるまで待つ

変数の条件待ちも可能な為、次の表現をとることができます。

ＷＡＩＴ　Ａ＝１

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

```
WAIT A<>B    ＡとＢが等しくない
WAIT A><B    ＡとＢが等しくない
WAIT A=B     ＡとＢが等しい
WAIT A>B     ＡよりＢが小さい
WAIT A<B     ＡよりＢが大きい
WAIT A=>B    ＡよりＢが小さいか等しい
WAIT A>=B    ＡよりＢが小さいか等しい
WAIT A=<B    ＡよりＢが大きいか等しい
WAIT A<=B    ＡよりＢが大きいか等しい
```

例)入力ポート１の信号に従って出力ポート２を制御しています。

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

【ＷＰＬＳ】　　機能：パルス　　種別：コマンド　　サポート：Ｐ

■書　式　WPLS ｐ ｒ ｃ

VPLS ｐ ｒ ｃ

ｐ：出力ポート（out0かＭＩＦ－８１６のパルスポート）

CW =1,4,16,64　(1=X,4=Y,16=U,64=Z)

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

ｒ：オフパルス幅　2.17μ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。いくつパルスを出したかを記録する変数

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

```
WPLS A,B0,C0
```

Ａはポート指定です。ＹＰＬＳと同様です。

Ｂ０はオフ時間を規定します。時間の単位は２．１７μ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。停止方法は停止待ちの方法はＹＰＬＳと同様です。

```
PLS_MIF                    ←出力をMIFに変更
FORK 1,*TASK1
```

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

PLS_MIF,YPLS参照

【ＷＳ０】　　機能：Ｉ／Ｏ　　種別：関数　　サポート：Ｐ・Ｚ

■書　式　WS0(n)
WS1(n)

n：ポート番号

0≦n≦255　　Ｉ／Ｏ
-128≦n≦-1　メモリーＩ／Ｏ(3.22L以上で使用可)

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

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

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

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

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

【ＷＳ１】　機能：Ｉ／Ｏ　種別：関数　サポート：Ｐ・Ｚ

WS0参照

【Ｘ】　機能：パルス　種別：配列　サポート：Ｐ・Ｚ

■書　式　X(n)

n：点番号

Ｐ版　0≦n≦300

Ｚ版　0≦n≦255

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

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

点３(Ｐ(３))のＸ，Ｙの値が１００，２００であることを表示しています。

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

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

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

【ＸＲＡＮＧ】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

■書　式　XRANG max,min

YRANG max,min

URANG max,min

ZRANG max,min

Ｐ版　-8388607≦min<max≦8388607

Ｚ版　-32766≦min<max≦32766

■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。Ｐ版では指定した値に対してレンジで２５６パルスの誤差を持ちます。

```
XRANG 10000,0
```

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

n'=(n/256)×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。

【Z版について】
ＸＲＡＮＧ，ＹＲＡＮＧは、ＪＯＧコマンドによる移動領域を制限するものです。
ＵＲＡＮＧ，ＺＲＡＮＧは、ＪＯＧＺＵコマンドによる移動領域を制限します。

【Y】　　機能：パルス　　種別：配列　　サポート：Ｐ・Ｚ

■書　式　Y(n)

n：点番号
Ｐ版　0≦n≦300
Ｚ版　0≦n≦255

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

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

点３(Ｐ(３))のＸ，Ｙの値が１００，２００であることを表示しています。

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

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

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

【ＹＰＬＳ】　　機能：パルス　　種別：コマンド　　サポート：Ｐ

■書　式　YPLS p,r,c
ZPLS p,r,c

p：出力ポート(out0かＭＩＦ－８１６のパルスポート)
CW =1,4,16,64　(1=X,4=Y,16=U,64=Z)
CCW=2,8,32,128 (2=X,8=Y,32=U,128=Z)
r：パルスレート　１０pps－１５００pps程度
c：パルスカウンター　省略可。いくつパルスを出したかを記録する変数

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

```
 10 A=1
 20 B=100
 25 C0=0
 30 YPLS A,B,C0
 40 FOR I=1 TO 10
 42 I0=100*I
 45 PRINT I0
 50 WAIT C0=I0
 55 A=0                    ←停止
 60 WAIT A=256             ←停止待ち
 65 A=1
 70 YPLS A,B,C0
 80 NEXT I
 90 A=0
>RUN
 100
 200
```

```
 300
 400
 500
 600
 700
 800
 900
 1000
>
```

PLS_MIF,WPLS参照

【ＹＲＡＮＧ】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

XRANG参照

【Ｚ】　機能：パルス　種別：配列　サポート：Ｐ・Ｚ

■書　式　Z(n)

n：点番号

Ｐ版　0≦n≦300

Ｚ版　0≦n≦255

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

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

点３(Ｐ(３))のＵの値が１００であることを表示しています。

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

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

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

【ＺＰＬＳ】　機能：パルス　種別：コマンド　サポート：Ｐ

YPLS参照

【ＺＲＡＮＧ】　機能：パルス　種別：コマンド　サポート：Ｐ・Ｚ

XRANG参照