第３章　プログラミング例

３．１　基本１　（ＩＯ定義）

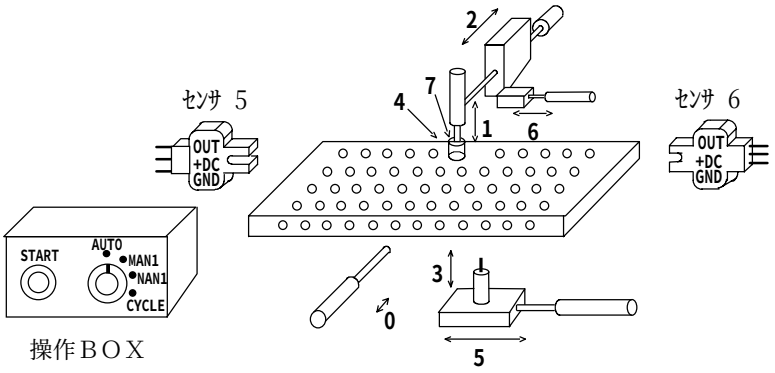
ＭＰＣの入出力はそれぞれ０番からスタートしています。ＭＰＣ－８１６に付属している１６点の入力と８点の出力は０～１５の入力、０～７の出力にふりあてられています。ＭＩＦではこれが１６～３１(入力)８～１５(出力)と順にふられています。詳細は巻末の諸元表を参照して下さい。ＭＰＣではＩ／Ｏをラベルで扱うことができます。これを定義するコマンドは＃ＤＥＦＳ、＃ＤＥＦＯです。それぞれ入力ポート、出力ポートの定義に用います。もちろんラベルを使用しないでそのままポート番号で扱うこともできますが保守時にはラベル化されたＩ／Ｏの方が扱いやすくなります。次に、定義例を示します。最初に０から６までの入力を指定し、次に０から７までの出力ポートを定義しています。この定義は最初にしなくても、プログラム完成後追加しても有効です。

```
#DEFS START 0
#DEFS AUTO 1
#DEFS MAN1 2
#DEFS MAN2 1
#DEFS CYCLE 4
#DEFS MIGI 5
#DEFS HIDARI 6
'
#DEFO PIN 0
#DEFO HANDDWN 1
#DEFO HANDMAE 2
#DEFO PINUP 3
#DEFO AIR 4
#DEFO MAE 5
#DEFO STOP 6
#DEFO VACUM 7
```

Ｉ／Ｏマップ

	ポートNO.	名　称	
入力	0	ＳＴＡＲＴ　スイッチ	押してＯＮ
	1	ＭＯＤＥ　スイッチ　ＡＵＴＯ	
	2	〃　　ＭＡＮ１	
	3	〃　　ＭＡＮ２	
	4	〃　　ＣＹＣＬＥ	
	5	テーブル右	さえぎってＯＦＦ
	6	テーブル左	〃
	7		
出力	0	位置決めピン	ＯＮ　位置決め
	1	トランスポート上下	ＯＮ　下降
	2	トランスポート前後	ＯＮ　前進（Ｃ）
	3	案内板送りピン上下	ＯＮ　上昇
	4	トランスポート真空破壊	ＯＮ　真空破壊
	5	案内板送り前後	ＯＮ　前進
	6	度当り切替え	ＯＮ　Ｂ位置
	7	トランスポート真空	ＯＮ　真空

※前後の前（前進）とはシリンダのロッドが伸びた状態



３．２　基本２　（自動・手動運転の切り替え）

ＭＰＣにはマルチタスクの機能があります。これは、複数のプログラムを同時に実行するものです。この機能を使用することにより動作の切り替えを簡単に実現することができます。もしマルチタスクを使用しないと自動と手動の動作切り替えはプログラムを複雑なものにしてしまいます。

```
 '*******
*LOOP
 '*******
 WAIT SW(START)=1
 WAIT SW(START)=0
 QUIT 1
 IF SW(AUTO)=1 THEN *A1
 IF SW(MAN1)=1 THEN *M1
*A1
 FORK 1,*AUTO
 GOTO *LOOP
*M1
 FORK 1,*MANU
 GOTO *LOOP
 '*******
*AUTO
 '*******
 FOR I=0 TO 7
 ON I
 TIME 5
 OFF I
 NEXT I
 GOTO *AUTO
 '*******
*MANU
 '*******
 FOR I=0 TO 7
 WAIT SW(MAN2)=1
 PRINT# I
 NEXT I
 GOTO *MANU
```

管理プログラム

自動運転プログラム
’プログラム自体には意味無し

手動運転プログラム
’プログラム自体には意味無し

この例ではＳＴＡＲＴスイッチの操作をきっかけにして、手動と自動を切り替えています。ＳＴＡＲＴスイッチ後ＡＵＴＯとＭＡＮ１のスイッチを読み取りこれにより必要に応じて異なるプログラムをＦＯＲＫしています。このＦＯＲＫコマンドがマルチタスクを起動させるものです。入力ポートＡＵＴＯがＯＮの場合このコマンド実行後、管理プログラムと自動運転プログラムが同時に稼働します。その次に、ＡＵＴＯをＯＦＦとしてＭＡＮ１をＯＮとしてＳＴＡＲＴスイッチを操作すると管理プログラム中のＱＵＩＴコマンドによりタスクが停止されます。もちろん動作をいきなり停止すると危険なプログラムもありますのでタスク間インターロックによってタイミングを考慮します。通常はサイクル停止となるようにします。このように動作を決めるプログラムと動作を管理するプログラムを分離することによって見通しの良いプログラムとなります。また、管理プログラム中で非常停止スイッチを監視するようにすればいつでもプログラムを停止したり特別な処理をすることができます。次の例では非常停止によってプログラム停止、出力ポートクリア(ＳＥＴＩＯ)後にプログラムを停止しています。

```
 '*******
*LOOP
 '*******
 IF SW(START)=1 THEN *NORMAL
 IF SW(EMG)=1 THEN *EMG
*NORMAL
 WAIT SW(START)=0
 QUIT 1
 IF SW(AUTO)=1 THEN *A1
 IF SW(MAN1)=1 THEN *M1
*EMG
 QUIT 1
 SETIO
 END
*A1
 FORK 1,*AUTO
 GOTO *LOOP
*M1
 FORK 1,*MANU
 GOTO *LOOP
```

管理プログラム

また、マルチタスクは管理プログラムだけでなく装置の上に分散する幾つものユニットに個別のプログラムを対応させて使用します。

３．３　基本３　（タスク間インターロック）

先の例のように管理プログラムが一方的にタスクを停止すると不都合なことが多いものです。このため、動作切り替えにはサイクル停止が必要になります。こうした場合はタスク(プログラム)間で信号のやりとりをして動作を定めます。

```
 '*******
*LOOP
 '*******
 WAIT SW(START)=1
 WAIT SW(START)=0
 ON -1
 WAIT SW(-2)=1              管理プログラム
 QUIT 1
 IF SW(AUTO)=1 THEN *A1
 IF SW(MAN1)=1 THEN *M1
*A1
 OFF -1,-2
 FORK 1,*MANU
 GOTO *LOOP
*M1
 ON -2
 FORK 1,*MANU
 GOTO *LOOP
 '********
*AUTO
 '*******
 FOR I=0 TO 7
 ON I
 TIME 5                     自動運転プログラム
 OFF I                      ’プログラム自体には意味無し
 NEXT I
 IF SW(-1)=0 THEN *AUTO
 ON -2
 WAIT SW(-1)=0
 GOTO *AUTO
```

ここでは、メモリＩ／Ｏの－１と－２を使用して＊ＡＵＴＯのプログラムの停止を制御しています。このように相手のプログラムを特定の箇所で停止させたり、状態を知らせたりするのにはメモリＩ／Ｏが有効です。ここでは直接番号でメモリＩ／Ｏを指定していますが＃ＤＥＦＳ、＃ＤＥＦＯで定義することもできます。注意すべきことはメモリＩ／Ｏは入出力が同じ番号のため、２つ定義文が必要となります。

```
#DEFS CMND -1
#DEFS ACK  -2
#DEFO CMND -1
#DEFO ACK  -2
```

３．４　基本４　（サブルーチン）

以上でプログラムを記述する場合の大まかな段取りができました。あと必要な事は細かい動作をどのように記述すれば良いかということです。これにはサブルーチンという考え方を使って、プログラムをよく整理します。もしこの整理が無いとプログラムは秩序の無い冗長なものとなって保守が難しくなります。次の例ではＰ＆Ｐの動作をさせるのに、動作をいくつかの単位に分けて全体を制御しています。Ｐ＆Ｐ１で１の位置に移動しＣＨＡＣＫでワークを囲みます。Ｐ＆Ｐ０は退避位置に移動するサブルーチンです。Ｐ＆Ｐ２は２の位置に移動します。その後ＲＥＬＳはワークを離します。この一連の動作をサブルーチンを使用しないで記述すると繰り返しが多いこと、全体の見極めがつきにくいことなどからプログラムの保守性が著しく悪くなります。またサブルーチン単体の動作確認はＲＵＮコマンド

で行います。

```
>RUN *P&P1
NO STATE!
>RUN *P&P0
NO STATE!
>
```

ＮＯ　ＳＴＡＴＥ！というエラーはサブルーチンのＲＥＴＵＲＮコマンドを実行した時に現れるものです。戻るべき文番号が無いために発生します。サブルーチンで注意すべきことはＲＥＴＵＲＮで戻らないでＧＯＴＯで戻してはならないことです。(必ずＲＥＴＵＲＮで戻して下さい)こうすると覚えておいた戻り番地が消費されないためにメモリ上にどんどん蓄積され３０回ほどでエラー停止を招きます。これまでのことをまとめるとプログラムはタスク、メインプログラム、サブルーチンによってよく整理して記述すべきだということです。モード切り替えなど全体に関わることはマルチタスクを使用して別のプログラムを起動するような見通しの良い構成にします。

```
*LOOP
 GOSUB *P&P1
 GOSUB *CHACK
 GOSUB *P&P0
 GOSUB *P&P2
 GOSUB *RELS
 GOSUB *P&P0
 GOTO *LOOP
```

```
*P&P1
 OFF HANDDWN,STOP
 TIME 50
 ON HANDMAE
 TIME 100
 ON HANDDWN
 TIME 100
 RETURN
```

```
*P&P2
 OFF HANDDWN
 ON STOP
 TIME 50
 ON HANDMAE
 TIME 100
 ON HANDDWN
 TIME 100
 RETURN
```

```
*P&P0
 OFF HANDDWN
 TIME 80
 OFF HANDME
 TIME 50
 RETURN
```

```
*CHACK
 ON VACUM
 TIME 80
 RETURN
```

```
*RELS
 OFF VACUM
 ON AIR
 TIME 20
 OFF AIR
 RETURN
```

３．５　パルス発生１　（バージョンの切り替え、モードの選択）

１）Ｐ版でのパルス発生

ＭＰＣ－８１６でのパルス発生は基本的にはＭＰＧ－３０３を使用します。またパルス発生の方法にはＭＯＤＥ５とＭＯＤＥ６があります。ＭＯＤＥ５はステップモータの駆動に用い、ＭＯＤＥ６はサーボモータに使用します。これは、多くのステップモータドライバが１０μから２０μsec以上のパルス幅を必要としていることに対して、サーボドライバは、１μsec程度に設定されていることに対応しているためです。パルス発生のパルスレートはＡＣＣＥＬコマンドで設定します。次の表はＡＣＣＥＬコマンドで与えた数値に対してどのような周波数でパルスが出力されるかを示しています。ＭＰＧでパルス発生させるためには最低次の様なコマンドが必要です。ここでは４相のステップモータで最大スピードを８kppsとしています。

```
        PG 1
        MODE 5
        OVRUN 0
        D45 0
        ACCEL 8000
        SETPOS 0 0
*LOOP
        MOVE 10000 0
        TIME 100
```

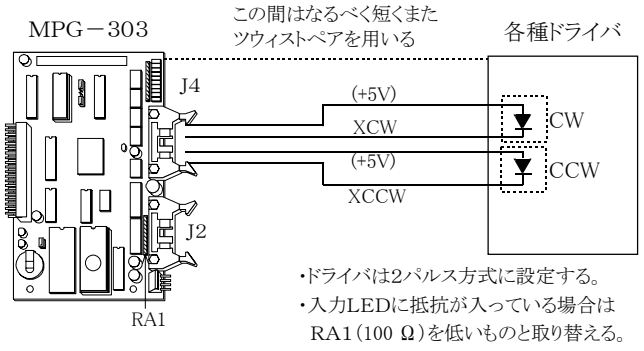
MPG-303ｶﾗﾊﾟﾙｽ発生				
REV 3.2(Pﾊﾞｰｼﾞｮﾝ)				
	MODE 5		MODE 6	
n	MOVE	JOG	MOVE	JOG
50000	--	--	56.3	14.6
45000	--	--	51.6	13.7
40000	--	--	44.2	12.0
35000	--	--	38.6	10.8
30000	29.0	13.7	32.5	9.3
25000	26.5	12.0	26.8	7.9
20000	21.1	8.9	21.3	6.4
15000	15.7	6.2	15.8	4.9
10000	10.4	3.9	10.3	3.3
8000	8.2	3.0	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	

（単位 kpps）

```
MOVE 0 0
GOTO *LOOP
```

最初に設定すべきコマンドはＰＧです。ＭＰＧ－３０３は、一つのシステムで３枚まで使用できます。このため、コマンドをどちらのＭＰＧに対して与えるかを最初に選択する必要があります。ＰＧ　１でショートピンが一番目に設定されたＭＰＧに対して有効となります。次にＭＯＤＥコマンドでモードを選択し、オーバランの設定をクリアしておきます。ＯＶＲＵＮが設定されているとＭＰＧのパルス動作が入力の状態によって停止します。Ｄ４５は直線補間の方法を決めるコマンドで通常の直線補間では０とします。コマンドの詳細は各コマンドリファレンスを参照して下さい。コマンドＳＥＴＰＯＳよりは実際の動作プログラムです。実際のプログラムではパルス発生に先立ち原点復帰をしますがこれは次の節にゆずります。コマンドＳＥＴＰＯＳは現在位置に指定した値を与えるコマンドです。これで現在位置を０，０として原点復帰に代えます。そのあとは１００００パルスの位置までステップモータを進め、タイマー１秒後にまた０の位置までもどります。このプログラムを実行するとＡＣＣＥＬの処で時間待ちが発生します。これはＡＣＣＥＬコマンドが加減速テーブルを内部計算で作成するために発生するロスタイムです。ACCEL 8000ではさほどの時間とはなりませんが５０kppsぐらいになると無視できなくなります。この時間は加減速の距離が長ければ長いほど多くなります。なおプログラム中でパルス発生のスピードを変更するにはＦＥＥＤコマンドを使用します。ＡＣＣＥＬで設定された範囲内で１５段階でスピード変更できます(FEED 0～FEED 15)。前記の実験をする前にＭＰＧとドライバの接続方法は次の様になります。ここではフォトカプラ入力のものを想定しています。ＴＴＬレベル入力や差動入力の場合はインターフェースが用意されていますので第２章「ＰＩＦ－４２２」を参照して下さい。

尚、ＭＰＧを使用した場合の原点復帰の入力ポートはＭＰＧ－３０３上のＪ２です。ＯＶＲＵＮの入力ポートはＭＩＦ－８１６上の入力ポート２４～３１となります。



２）Ｚ版でのパルス発生

ＭＰＣのパルス発生には他にＺ版、Ｓ版でのパルス発生があります。Ｚ版では外付けボード無しでＭＩＦ－８１６のＪ５よりパルスを出力します。このため低コストで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する、発生パルス数が±３２７６７に限定されているなどの制限があります。Ｚ版でのパルス発生モジュールはＭＯＤＥ１～ＭＯＤＥ４の４種類です。出力パルスレートは次の通りです。

MIF-816ｶﾗﾊﾟﾙｽ発生　(単位 Kpps)								
REV 2.20a								
	MODE 1		MODE 2		MODE 3		MODE 4	
n	MOVE	JOG	MOVE	JOG	MOVE	JOG	MOVE	JOG
32000	31.6	12.5	33.1	17.1	62.0	20.5	31.2	11.6
30000	29.5	11.6	31.1	15.5	57.6	19.0	29.3	10.8
25000	24.8	9.4	25.0	11.4	47.0	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	3.5	10.1	3.8	18.6	6.2	9.9	3.4
8000	8.0	2.8	8.1	2.9	14.7	4.9	7.8	2.7
4000	4.0	1.4	4.0	1.4	7.3	2.4	4.0	1.3
2000	2.0	0.7	2.0	0.7	3.7	1.2	2.0	0.7
ﾊﾟﾙｽ幅	8 μ sec		15 μ sec		最小 8 μ sec		10 μ sec	

ＭＯＤＥ１	ＸＹＵ３軸同時移動とＺ軸をサポートし動作範囲±３２７６７の範囲ながら直交４軸ロボットをサポートします。ただし途中停止をすると現在位置不明となり再度原点復帰しなければなりません。
ＭＯＤＥ２	ＸＹ２軸＋Ｚ軸サポートします。動作範囲は±３２７６７ですが、ＭＯＶＥコマンドでは６５５３６パルスの移動も可能です。途中停止の場合でも現在位置が保存されています。このモードがＭＰＣＩＮＩＴ後の標準状態です。
ＭＯＤＥ３	上記と同様の仕様ですが指定した値に対して約２倍の周波数を出力します。 ５０ kpps 程度を要求される小型のサーボに最適です。
ＭＯＤＥ４	１軸パルス発生です。出力ポートをＡＸＩＳコマンドで切り替えることができます。 ４軸パルス発生でそれぞれ排他的に動作する場合に経済的なモードです。

Ｚ版のパルス発生を使用するにはシステムローダーでＺ版に移行します。そしてＭＰＣＩＮＩＴで初期化します。ＭＰＣＩＮＩＴはプログラム中に記述してはいけません。その後は先のＰ版のプログラムと殆ど同じです。

(システムローダーでＺ版に変更)

```
>MPCINIT
>10 MODE 2
20  OVRUN 0
30  D45 0
40  ACCEL 8000
50  SETPOS 0 0
60  *LOOP
70  MOVE 10000 0
80  TIME 100
90  MOVE 0 0
100 GOTO *LOOP
```

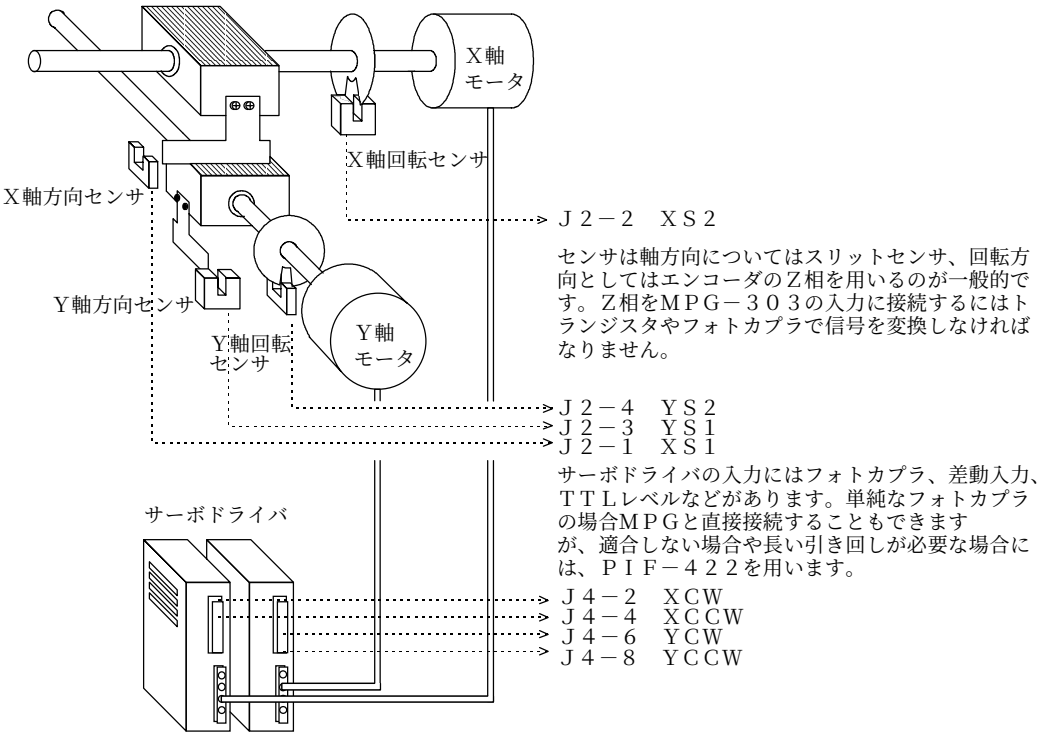
＊Ｚ版パルス発生でのＭＩＦの信号の扱い

入力１６	ＸＳ１	Ｘ軸用	入力２４	ＯＶＲＵＮ１
１７	ＸＳ２		２５	２
１８	ＹＳ１	Ｙ軸用	２６	３
１９	ＹＳ２		２７	４
２０	ＵＳ１	Ｕ軸用	２８	５
２１	ＵＳ２		２９	６
２２	ＺＳ１	Ｚ軸用	３０	７
入力２３	ＺＳ２		入力３１	ＯＶＲＵＮ８

Ｚ版での原点復帰センサの入力ポートはＭＩＦ－８１６の１６～２３の入力ポートとなります。また途中停止や非常停止のポートは同じくＭＩＦ上の入力２４～３１です。

３．６　パルス発生２　　（ＸＹステージ・原点復帰・ティーチング・パレタイズ）

パルス発生を用いた応用の中で、比較的難しくなるのがＸＹステージの制御です。これはＸＹステージではロボット的な動作を要求されることが多くなるためです。その一番の応用例はパレタイズです。ここでは次の様な簡単な応用例にもとづいて解説します。



１）原点復帰について

原点復帰は通常ふたつのセンサによって実施します。一つは実際のテーブルの移動方向に取り付けられたスリットセンサ、もう一つは回転軸に取り付けられたスリットセンサです。サーボモータの場合はＺ相検出を用いるかサーボドライバ自体に備えられた原点検出機能を使用します。ＭＰＣで原点復帰する場合には次のふたつのコマンドが重要です。

```
SHOM 2 8 10
HOME &HA 100 100
```

ＳＨＯＭコマンドは原点復帰のパルス発生の方向とパルス発生のスピードを定めます。前記の様な設定例ではＸＹ軸ともにＣＣＷ方向に原点復帰パルス出力します。逆にＣＷで原点復帰する場合は

```
SHOM １ 4 10
```

となります。ＳＨＯＭの最後の引き数は原点復帰パルスのパルスレートで、ＡＣＣＥＬで設定されたパルスレートとは独立して設定します。ＭＯＤＥ５，６での原点復帰パルスレートは次の様に計算します。

f=8Mhz/(1841+208×n)

例えば前記の様に１０とすれば２０４０．３となり約2kHzとなります。またパルス幅はデューティ５０%となっています。ＨＯＭＥの最初の引き数は原点復帰入力の目的とするビットパタンを表しています。通常、遮光センサは常時ＯＮでスリットの遮光でＯＦＦとなります。またサーボのＺ相はＺ相検出でＯＮとなります。次の表の通り、HOMEの最初の引き数は目的とするセンサパタン値(原点復帰後の最下段)を与えます。尚、この値は関数ＨＰＴ(０)で直接読み取ることができます。関数ＨＰＴ()はＭＰＧ動作中には使用できません。ＨＯＭＥの２番目と３番目の引き数は原点復帰に先立つ退避移動量です。この例の様にＨＯＭＥ　＆ＨＡ　１００　１００とすると原点復帰前にCW方向に１００パルスずつ移動します。これは原点復帰時には一定の方向から原点復帰をしないと一定の精度を確保できなかったり、Ｚ相検出(Ｃ相)が一回転ズレてしまうことがあるためです。必要の無い場合は0、0とします。又、Ｚ相検出(Ｃ相)ではパルス幅や動作の不安定さから検出ミスすることがあります。これを防ぐのにＳＨＲＤコマンドで６～１５までの値を与えます。電源投入後は４となっています。この場合の原点復帰のパルスレートは次の通りです。

Ｊ２	方向	センサ	原点復帰前	原点復帰後
1 XS1	X方向	遮光センサ	ON (1)	OFF (0)
2 XS2	X回転	Z相	OFF (0)	ON (1)
3 YS1	Y方向	遮光センサ	ON (1)	OFF (0)
4 YS2	Y回転	Z相	OFF (0)	ON (1)
-	-	-	&H5	&HA

f=8MHZ/(709+RX(n×52+283))

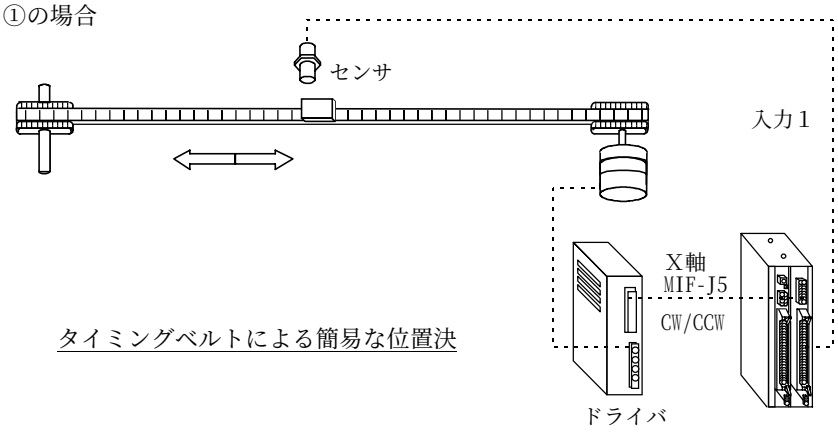
＊ＲはＳＨＲＤの設定値。ｎはＳＨＯＭの設定値
＊Ｚ相はメーカによってＣ相と表記されることもある。

☆Ｚ版での原点復帰例

ＨＯＭＥコマンドでは入力ポート１６～２３をビットパタンとして条件停止します。原点復帰には概ね次の３種類が考えられます。この３つの場合に準じて適切なコマンドを実行します。

①簡易単軸制御の場合のような、１センサによる原点位置決め
②ステップモータなどを利用した単軸あるいはＸＹテーブルで回転軸の検出を含む場合
③サーボモータなどを利用した単軸あるいはＸＹテーブル

①の場合



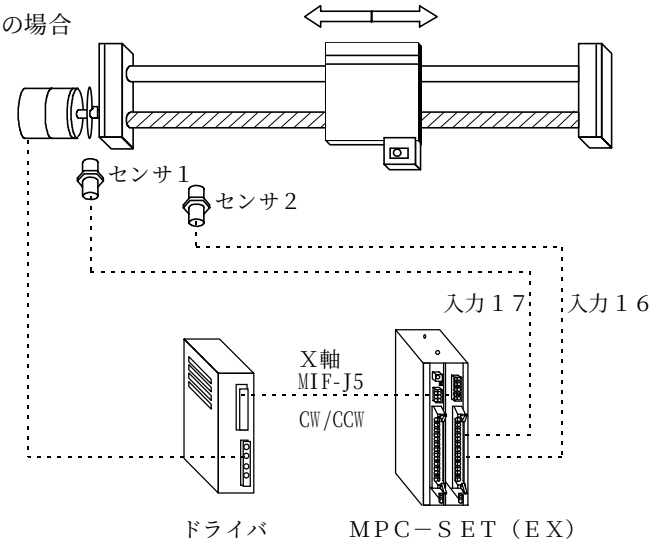
タイミングベルトによる簡易な位置決

図の様に入力ポート１６にセンサを結合しX軸のパルス発生を使用する場合の原点復帰は、次の様なコマンドで実施します。

```
10 SHOM 2,0,10
20 HOME 1
```

ここではセンサがＯＮで原点復帰成立としています。ＳHOMの３番目の１０はスピード設定です。必要に応じて変更して下さい。ＳHOMで２番目の引き数が０なのはＹ軸の原点復帰は実施しないということです。HOME　１は入力ポート１６がＯＮで条件成立という意味です。センサが条件成立でＯＦＦの場合はHOME　０とします。

②の場合



ボールネジスライダーによる１軸制御

この場合のステップモータは、リードネジやボールネジが使用されます。又、原点センサは移動方向に１つ回転軸に１つとなります。原点の条件としてはセンサ２とセンサ１が共にＯＦＦとなった場合などがあります。

```
10 SHOM 2,0,10
20 HOME 0
```

この例では入力ポート１６と１７がＯＦＦとなるのを待ちます。ＯＮになるのを待つのであれば

```
20 HOME 3
```

ここでもし２軸のX，Yテーブルであれば次の様にします。

```
10 SHOM 2,8,10
20 HOME &HF (ｱﾙｲﾊ HOME 0)
```

③の場合

サーボモータの原点復帰では概ね次の様に分類されます。

（１)サーボ軸のリミットセンサとサーボモータエンコーダのＺ相(Ｃ相とも呼ばれる)とのＡＮＤ条件

（２)サーボドライバに備えられた原点復帰機能に頼る場合(オムロンのサーボドライバ等)

(１)の場合には次の注意が必要です。

まずＺ相(Ｃ相)のパルス幅が非常に細いのでＳＨＲＤコマンドによりセンサ検出機能を感度の良いものとしなければなりません。ＳＨＲＤのデフォルトは４です。通常この値は　ＳＨＲＤ　６　～　ＳＨＲＤ　１５　までです。ＳＨＲＤの値は大きい程感度は良くなるのですが、原点復帰のスピードは遅くなります。さらに、(１)の場合に早いスピードで原点復帰した時と遅いスピードで原点復帰をした時とでは停止位置が微妙に変わります。これは、サーボ特有のたまりパルスの影響によるものです。正確な原点復帰を期待する場合はゆっくりとした原点復帰が必要となります。

(２)の場合

原点復帰動作指示入力を与えておけばたまりパルスの影響や、Ｚ相のパルスの細かさも問題となりませんので精度の良い原点復帰ができます。

☆原点復帰の注意

ＭＰＣの原点復帰はX，Ｙ，Uの３軸をサポートするＨＯＭＥ命令、Ｚ軸の原点復帰を扱うＨＯＭＺ命令があります。ＨＯＭＥ命令はMＯＤＥ１，２，３とも共通の仕様となっています。
使用上では次の点に留意して下さい。

(１)ＨＯＭＥコマンドは入力ポート１６～２１の入力を同時に監視しています。このためX軸のみの原点復帰でも入力１８，１９，２０，２１の入力が不定となるとＹ軸，U軸の原点復帰が実施され、ＨＯＭＥコマンドから抜け出せなくなることがあります。入力１６～２１はＺ版を使用した場合です。ＭＰＧを使用した場合は、Ｊ２の１～６までのＸＳ，ＹＳ，ＵＳの各原点入力です。

(２)ＨＯＭＥコマンドと関わりあうのは次のコマンドです。

ＳＨＯＭ　ｘ，ｙ，ｎ	ｘ，ｙ原点復帰指示とＨＯＭＥとＨＯＭＺのスピード
ＳＨＭＺＵ　ｚ，ｕ	ｚ，ｕ軸の退避量設定
ＨＯＭＥ　ｎ[，ｘ，ｙ]	ｘ，ｙ，ｕ軸の原点復帰
ＨＯＭＺ　ｎ	ｚ軸の専用原点復帰
ＳＨＲＤ　ｎ	原点復帰の検出感度

ＭＰＣでは２軸サポートから４軸に拡張された時の形式を引き継いでいるために、命令体系がよく整理されていません。(＾＾；)

２）ティーチング

点データにはあらかじめ設定してある場合と、装置が組上がってから設定するべきデータの２通りがあります。次に点データの扱い方について例示しますがその前に点データ(ベクトル)配列Ｐ(ｎ)と成分配列Ｘ(ｎ)，Ｙ(ｎ)，Ｚ(ｎ)，Ｕ(ｎ)の関係について解説します。ＭＰＣには予約配列としてＡＲ(ｎ)は純粋に演算用の配列ですが、他は点データとしての意味をもっています。点データはＰ(ｎ)であらわされ、その各成分はＸ(ｎ)，Ｙ(ｎ)，Ｚ(ｎ)，Ｕ(ｎ)で表されています。点配列Ｐ(ｎ)は通常の演算で使用することができません。受け付けるコマンドはＭＯＶＥ，ＪＭＰＺ，ＪＵＭＰ，ＰＲＩＮＴの点データに基づく移動コマンドと表示コマンドのみです。これに対してＸ(ｎ)～Ｕ(ｎ)は予約配列として単体で演算中で使用することができます。

①点データはコマンドＳＥＴＰ，ＳＴＰＺＵで設定できる。

```
>SETP 1 100 200
>STPZU 1 300 400
```

②点データは各配列成分に演算で設定できる。

```
>X(1)=100
>Y(1)=200
>Z(1)=300
>U(1)=400
```

③点データはプログラム中で設定できる。

```
>10 SETP 1 100 200
20 STPZU 1 300 400
```

④点データはプログラム中で変更できる。

```
>10 X(1)=X(1)+D0
20 Y(1)=Y(1)+D1
```

点データはこのようにして様々な方法で設定・変更できます。またこの使用方法は様々です。次の３つの例はいずれも同等の意味を持ちます。

```
① MOVE P(1)
② MOVE X(1) Y(1)
```

```
③ A0=X(1)
   A1=Y(1)
   MOVE A0 A1
```

もうひとつの点データの設定方法は、ティーチコマンドによるものです。ティーチコマンドはインチング動作(ＪＯＧ動作)により機械体を目的の位置まで移動させ、その点を教示するものです。ティーチモードに移るにはＴ＜ＥＮＴＥＲ＞とします。単純な初期化が終わった時点では、次の様にデタラメな値が表示されますので所定の初期化を実施します。ティーチモードに移って最初に押すキーは０～３のいずれかです。このキーによってインチング量を選択します。実際に移動させるコマンドは、ＸでＣＷ，ｘでＣＣＷです。これはX軸の場合で、他の軸の場合はＹ，Ｚ，Ｕキーです。目的の場所への移動が終了したら、Ｐを押します。すると改行して番号の入力を促してきます。ここで指定された番号のＰ(ｎ)に現在位置が登録されます。次の例では点１と点３にデータを設定し、ティーチモード終了後ＰＲコマンドで表示しています。尚、ティーチモードの終了はＱです。

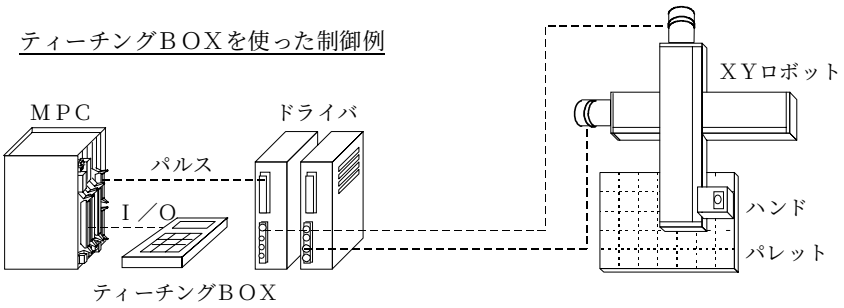
```
>MPCINIT
PG 2#(X,Y,Z,U) 2048 28691 2048 28691 [XYZ,U] 400 400 （ここでT<ENTER>を実行しています）
>PG 1
>MODE 5
>ACCEL 15000
>SETPOS 0 0
>STPZU 0 0 0
PG 1#1(X,Y,Z,U) 0 0 0 0 [XYZ,U] 200 200      （ここでもT<ENTER>を実行しています）
>
PG 1#1(X,Y,Z,U) 400 0 0 0 [XYZ,U] 200 200
P1
PG 1#1(X,Y,Z,U) 800 0 0 0 [XYZ,U] 200 200
P3
PG 1#1(X,Y,Z,U) 800 0 0 0 [XYZ,U] 200 200
>PR P(1) P(3)
 400 0 0 0 800 0 0 0
```

主なティーチモードでの操作キャラクタは次の通りです。　　参照コマンド：Ｔ

Ｘ ｘ	Ｘ軸のインチング移動(小文字でマイナス方向)
Ｙ ｙ	Ｙ軸のインチング移動(小文字でマイナス方向)
Ｚ ｚ	Ｚ軸のインチング移動(小文字でマイナス方向)
Ｕ ｕ	Ｕ軸のインチング移動(小文字でマイナス方向)
０１２３	インチング量の選択
Ｐ	点データの設定
Ｑ	ティーチモードの終了

３）ティーチングプログラム

装置にティーチングタブレットを装備してティーチング機能を備えることは多くあります。Ｉ／Ｏに次の図の様にスイッチボックスを接続してＸＹステージなどをならい教示させるものです。このティーチングにはＪＯＧ命令を使う方法とＲＭＯＶによって構成する場合の２通りがあります。



☆ＲＭＯＶを使用する場合

ＲＭＯＶは現在位置よりの相対移動です。スイッチ入力によって、ＲＭＯＶを実行しインチング動作させます。点の教示ではＳＥＴＰコマンドを使用します。次のプログラムはX軸１軸の場合の例です。インチング量を変更する場合はＲＭＯＶの引き数を変数として、変数をデジスイッチで設定するようにします。また、点番号の設定も同様な方法をとるものとします。

```
1000 IF SW(0)=1 GOSUB 1100
1010 IF SW(1)=1 GOSUB 1200
1020 IF SW(2)=1 GOSUB 1300
1030 GOTO 1000
1100 RMOV 10,0
```

```
1110 RETURN
1200 RMOV -10,0
1210 RETURN
1300 SETP 1,X(0),Y(0)
1310 RETURN
```

☆ＪＯＧ命令を使用する場合

①ＭＰＧ－３０３を使用する場合

ＭＰＧでのＪＯＧ機能はＪＯＧコマンドとＸＲＡＮＧ～ＵＲＡＮＧまでの領域設定コマンドより成立しています。ＪＯＧコマンドはＳＴＯＰコマンドで停止となります。次にＸ１軸のみでＪＯＧコマンドを使用した例を示します。

```
910 XRANG 10000,-10000
1000 IF SW(0)=1 GOSUB 1100
1010 IF SW(1)=1 GOSUB 1200
      |
1100 JOG 100,1
1110 WAIT SW(0)=0
1120 STOP 1
1125 TIME 5
1130 RETURN
1200 JOG 100,2
1210 WAIT SW(1)=0
1220 STOP 1
1225 TIME 5
1230 RETURN
```

点の教示などは先の例と同様なので省略してあります。動作範囲の設定はＸＲＡＮＧ(ＹＲＡＮＧ，ＵＲＡＮＧ，ＺＲＡＮＧ)で定めておきます。また、STOP後のTIME 5はＭＰＧ終了待ちです。

②Ｚ版でのＪＯＧ

Ｚ版ではＪＯＧ，ＪＯＧＺＵコマンドを使用します。Ｐ版と使い方が異なっていますので注意して下さい。ＪＯＧ，ＪＯＧＺＵでの移動のリミットは、ＸＲＡＮＧ～ＵＲＡＮＧまでの設定で使用することができます。

```
100 XRANG 10000,0
110 YRANG 10000,0
120 IF SW(4)=1 GOSUB 1000
130 IF SW(5)=1 GOSUB 1000
140 IF SW(6)=1 GOSUB 1000
150 IF SW(7)=1 GOSUB 1000
160 GOTO 120
     |
1000 JOG 500
1010 RETURN
```

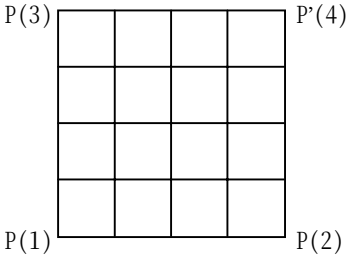
ＪＯＧコマンドは、ボタンを押した状態で連続移動する場合に用いられるティーチング機能です。ｎは加速距離で、これはＡＣＣＥＬで指定した加速距離より小さい値を設定します。その範囲での数値が大きい程スピードは早くなります。Ａ１はポートバンク指定ですが、省略すると&H48(ＣＰＵボードの入力ポート)として扱われます。ＪＯＧがＭＳＢ側、ＪＯＧＺＵがＬＳＢ側にアサインされそれぞれ独立にアドレスを指定することができます。

ビット	B7	B6	B5	B4	B3	B2	B1	B0
ポート番号	7	6	5	4	3	2	1	0
制御軸	-Y	+Y	-X	+X	-Z	+Z	-U	+U

４）パレタイズについて

点データの中には、パレットのように大量の点を必要とするものがあります。これを前記の様に点データの教示によって実現しようとするとたいへんな労力となります。パレタイズコマンドはこのような規則的な点データを少ない基本的な点データから生成するものです。次のプログラム例では点１、２、３を基本点とする５×５のパレットの点を定義し、データは関数ＰＬ(ｎ)で取り出しています。

```
100 PALET 1 2 3
110 MTRX 5 5
120 FOR I=1 TO 25
130 MOVE PL(I)
140 NEXT I
```



P(1)=PL(1)
P(2)=PL(5)
　　|
P(3)=PL(21)
P'(4)=PL(25)

パレットはP版の場合同時に２つ設定することができます。コマンドはＰＡＬＥＴ１とＭＴＲＸ１です。対応するパレット関数はＰＬ１(ｎ)です。さらに点データと同じようにパレット点はX、Yの各成分で得ることができるように関数ＰＬＸ()，ＰＬＹ()が用意されています。尚、パレット点ＰＬ(ｎ)はX，Y，Z，Uの４成分を含んでいますがパレタイズされているのはX，Yの２次元のみでU，Z成分についてはＰＡＬＥＴコマンドの最初の点の座標値にならいます。前記の例では、P(1)のZ,U値が各PL(n)のU,Z成分となります。

５）実際のプログラム

ずいぶんと前置きが長くなりましたが、これで実際の応用の準備ができました。次にモデルプログラムを例示します。自動、ティーチング、原点復帰を単純なモデルで表現しています。流れと基本的なコマンドをつかんで下さい。

```
#DEFS START 1
#DEFS HOMES 2          メイン操作スイッチ
#DEFS TEACH 4
'
#DEFO READY 1
#DEFO ERR 2
'
#DEFS SNS1 5   <------ インタロック用センサ
#DEFS XCW 6
#DEFS YCW 7            各軸のＪＯＧキー
#DEFS XCCW 8
#DEFS YCCW 9
#DEFS PENT 10  <------ 現在位置記憶スイッチ
#DEFS OP1 11   <------ パレット交換完了スイッチ
#DEFO PALET 5  <------ パレット交換表示
'************
'INIT
'************
PG 1                   ＭＰＧ－３０３初期化
MODE 5                 この例のようにＰＧ選択、モード設定、ＯＶＲＵＮ、
OVRUN 0                Ｄ４５等を初期してからＡＣＣＥＬを実行する
D45 0
ACCEL 15000
PALET 11,12,13         パレット宣言点１１，１２，１３を使って５×５のパレット構成
MTRX 5,5
'************
*MAIN
 '************
 IF SW(START)=1 GOSUB *START     メインプログラム
 IF SW(HOMES)=1 GOSUB *HOME      自動、原点復帰、ティーチングの動作を分類する
 IF SW(TEACH)=1 GOSUB *TEACH
 GOTO *MAIN
 '************
*START
 '************
 MOVE P(1)
 ON PALET       <------ 出力パレットのＯＮはオペレータにパレット交換を促す
 WAIT SW(OP1)=1 <------ ＯＰ１はオペレータスイッチで作業完了を意味する
 OFF PALET
 J=0
```

```
*LOOP
 WAIT SW(SYS1)=1
 MOVE PL(J)
 GOSUB *PLACE
 IF J=25 THEN *END
 GOTO *LOOP
*PICK
 'PICK MOTION
 RETURN
*PLACE
 'PLACE MOTION
 RETURN
*END
 RETURN
 '************
*HOME
 '************
 SHOM 2,8,10
 HOME &H000A,100,100
 RETURN
 '************
*TEACH
 '************
 IF SW(XCW)=1 THEN *XMV
 IF SW(YCW)=1 THEN *YMV
 IF SW(XCCW)=1 THEN *XMVN
 IF SW(YCCW)=1 THEN *YMVN
 IF SW(PENT)=1 THEN *PENT
 GOTO *TEACH
*XMV
 JOG 100,1
 WAIT SW(XCW)=0
 GOTO *MV_END
*XMVN
 JOG 100,2
 WAIT SW(XCCW)=0
 GOTO *MV_END
*YMV
 JOG 100,3
 WAIT SW(YCW)=0
 GOTO *MV_END
*YMVN
 JOG 100,4
 WAIT SW(YCCW)=0
*MV_END
 STOP 1
 TIME 10
 GOTO *TEACH
*PENT
 A=IN(2)^&H000F
 A1=IN(2)/16
 A1=A1*10
 A=A+A1
 SETP A,X(0),Y(0)
 RETURN
```

パレットとパルス発生の組み合わせ例
移動スピードを変更する場合は’ＦＥＥＤ’を使用します

チャッキング動作

プレース動作

原点復帰動作
サーボのＺ相を読み取る場合は他にＳＨＲＤを必要とします
（ＳＨＲＤ　６～ＳＨＲＤ　１５）

ティーチングスイッチによる分岐

各軸のＪＯＧコマンドと停止
停止はスイッチを離す（開放）ことによって行っています。
スピードを変更したい場合はＪＯＧの１００を変更します。

ＪＯＧコマンドはＳＴＯＰによって停止します
停止後０．１秒程のタイマを入れます

ティーチング点はデジスイッチで指定することを前提としています
入力１６～２３にＤＳＷが２ヶあると想定しています

３．７　パルス発生３

パルス発生の作業の中で最もやっかいなのがパルス発生中の途中停止です。パルス発生はＩ／Ｏ制御と異なり停止が必要となった時点でどのような状態であるか、あるいはどのように停止したいかによって処理が変わります。残念ながらＭＰＣ－８１６での途中停止は仕様追加によって構成されているためわかりにくいものであったり、コマンドが対称的にサポートされていなかったりします。この節では代表的なパルス停止の方法を紹介します。

１）パルス停止の注意事項

☆ＱＵＩＴの禁止

マルチタスクの使用方法に習熟してくると、ある作業の停止にはＱＵＩＴが便利だということがわかりますが、これは、パルス発生の場合にはあてはまりません。パルス発生中のタスクに対して不用意にＱＵＩＴするとそれ以降ＭＰＧに対してのコマンドが無効になることがあります。

悪い例

```
*LOOP
 FORK 1 *PG
 TIME 100
 QUIT 1
 TIME 100
 GOTO *LOOP
*PG
 RMOV 1000 1000
 TIME 50
 RMOV -1000 -1000
 TIME 50
 GOTO *PG
```

良い例

```
*LOOP
 FORK 1 *PG
 TIME 100
 STOP 4
 WAIT BSY(1)<>0
 QUIT 1
 STOP 5
 GOTO *LOOP
*PG
 RMOV 1000 1000
 TIME 50
 RMOV -1000 -1000
 TIME 50
 GOTO *PG
```

☆ＳＴＯＰ後の処理（ＳＴＯＰさせても次の命令は有効）

ＭＰＧにＳＴＯＰをかけてもインタプリタの停止が止まるわけではありません。パルス発生中のコマンドが停止させられるだけです。このため、ＳＴＯＰによって完全に動作を中断させるには次の様な工夫が必要です。

悪い例

```
   FORK 1 *MPG
   WAIT SW(1)=1
   STOP 1
   END
*MPG
   RMOV 100 0
   GOTO *MPG
```

良い例

```
FORK 1 *MPG
WAIT SW(1)=1
STOP 1
END
*MPG
 RMOV 100 0
 WAIT BSY(0)=1
 GOTO *MPG
```

悪い例ではRMOV 100 0が１回停止させられてもすぐ次のコマンドが実行されて、再びRMOV 100,0が実行されます。良い例ではＲＭＯＶごとにWAIT BSY(0)=1のチェックをしており１度ＳＴＯＰで停止させられるとここで条件待ちとなります。

☆ＪＯＧ，ＳＴＯＰ後の現在位置読み取り

ティーチング用のプログラムを作る場合にＪＯＧとＳＴＯＰを組み合わせますが、ＳＴＯＰ実行後ただちに現在位置を読み取る関数を実行すると、ＳＴＯＰが無効になります。次に１軸のティーチングプログラム例を示します。ＳＴＯＰ後に０．１秒のタイマを入れています。これはＳＴＯＰの実行とＭＰＧの停止の間には時差があり、この間にＭＰＧに対して指示を与えると誤動作してしまうことがあるためです。

```
*TEACH
 IF SW(1)=1 THEN *XCW
 IF SW(2)=1 THEN *XCCW
 IF SW(3)=1 THEN *END
 GOTO *TEACH
*XCW
 JOG 100,1
 WAIT SW(1)=0
 STOP 1
```

```
 TIME 10                          ０．１秒タイマ
 X=X(0)
 GOTO TEACH
*XCCW
 JOG 100,2
 WAIT SW(2)=0
 STOP 1
 TIME 10                          ０．１秒タイマ
 X=X(0)
 GOTO *TEACH
```

このことは通常のＲＭＯＶやＭＯＶＥでも同様です。次の例のようにＳＴＯＰ停止をかけたあとでＭＰＧの実際の停止をＢＳＹ()関数によって監視する必要があります。

悪い例

```
PG 1
FORK 1,*PG
TIME 100
STOP 1
X1=X(0)
END
*PG
MOVE 10000,0
END
```

良い例

```
PG 1
FORK 1,*PG
TIME 100
STOP 1
WAIT BSY(1)<>0
X1=X(0)
END
*PG
MOVE 10000,0
END
```

☆複数のタスクからのコマンド

プログラムの設計時には各タスクの用途は整理されて、パルス発生用、Ｉ／Ｏ制御用と分類されていますが最終的な段階ではこの秩序が保ちきれなくなることがあります。例えば、前記の例にあったように他のタスクが停止コマンドを実行するために座標読み取りは他のタスクが実施します。又、複数のユニットがからみあった位置決めでは複数のタスクから同一のＭＰＧに対して指示を与えたくなることがあります。こうした場合には、交通整理をきちんとしなければなりません。セマフォを使う場合(複数のタスクが同じＭＰＧにコマンドを出力)次の様にセマフォ関数を使えば複数のタスクがコマンドを出しても問題は発生しません。

①セマフォの場合

```
*TASK1
 WAIT RSV(-1)=0
  MOVE 1000,0
 OFF -1
 GOTO *TASK1
```

```
*TASK2
 WAIT RSV(-1)=0
  RMOV -1000,0
 OFF -1
 GOTO *TASK2
```

②パルス発生を１つのタスクにまとめた場合

```
*TASK1
  A=1
  WAIT A=0
  GOTO *TASK1
```

```
*TASK2
 B=1
 WAIT B=0
 GOTO *TASK2
```

```
*PG
   IF A=1 THEN *MV
   IF B=1 THEN *RV
   GOTO *PG
*MV
   MOVE 1000,0
   A=0
   GOTO *PG
*RV
   RMOV -1000,0
   B=0
    GOTO *PG
```

２）ＳＴＯＰコマンド

ＳＴＯＰコマンドは１～５までの引き数があり次の様に意味が異なります。

STOP 1　減速停止　　ＳＴＯＰ後減速しその後停止します。

STOP 2　急停止　　ＳＴＯＰ後ただちにパルス発生中止します。

STOP 3　　次に続くパルス発生コマンドに対してのみ有効なＩ／Ｏ監視停止命令です。

```
STOP 3,10,1
MOVE 10000,0
```

この例ではポート１０がＯＮになったら次のＭＯＶＥコマンドで減速停止します。

STOP 4　　ＭＰＧコマンド受付を一時停止します。

STOP 5　　ＳＴＯＰ４による一時停止を解除します。

パルス発生を停止するコマンドは、この他にＯＶＲＵＮがあります。ＯＶＲＵＮは常時入力ポートを監視して条件が成立するとパルス発生を中止し、インタプリタの実行を停止します。このためエラー回復が必要な場合はＳＴＯＰ１，ＳＴＯＰ　２を監視タスクから実行します。又、ＳＴＯＰ　４，ＳＴＯＰ　５はパルス発生コマンドそのものを停止することができるのでこれを組み合わせると非常停止は簡単に実現することができます。

```
      FORK 1 *PG
*MAIN
      IF SW(1)=1 THEN *EMG
      IF SW(2)=1 THEN *JOB1
         |  その他の処理
      GOTO *MAIN
*EMG
      STOP 4                    ＭＰＧへのコマンド実行停止設定
      STOP 1                    ＭＰＧのパルス発生中止
      WAIT BSY(1)<>0            停止確認
         |
         |  後処理
      QUIT 1                    タスクの停止
      STOP 5                    ＭＰＧ停止の解除
      GOTO *EMG
*PG
      MOVE 1000 0
      WAIT SW(3)=1
      MOVE 0 0
      WAIT SW(4)=1
      GOTO *PG
```

もしＳＴＯＰ　４を先に実行しておかないと、ＳＴＯＰ　１はその時実行されているパルス発生コマンドのみに有効ですからパルス発生中止後すぐに次のコマンドを実行してパルス発生してしまいます。もう一つの方法は、各移動コマンドにＷＡＩＴ　ＢＳＹ(０)＝１を組み込んでおきます。

```
    FORK 1 *PG
    TIME
    STOP 1
    WAIT BSY(1)<>0
    QUIT 1
    END
*PG
    RMOV 1000,0
    WAIT BSY(1)=1               ＳＴＯＰで停止すると次へは進まない
    GOTO *PG
```

３）Ｚ版でのパルス中止

Ｐ版でのＭＰＧの停止はＳＴＯＰとＯＶＲＵＮを組み合わせて使用しますが、Ｚ版では大きく異なっています。ＯＶＲＵＮはコマンドの入力方法を除けばよく似ていますがＳＴＯＰに相当するコマンドはありません。これは、Ｚ版ではパルス発生中にマルチタスクが使えないためです。

ＭＯＤＥ１	途中停止はＯＶＲＵＮのみ有効です。
ＭＯＤＥ２，３	ＭＯＶＥ　Ｘ，Ｙ［，Ｊ］ ＲＭＯＶ　Ｘ，Ｙ［，Ｊ］ ＭＯＶＺ　Ｚ［，Ｊ］ ＲＭＶＺ　Ｚ［，Ｊ］ ＊Ｊが停止条件となります。
ＭＯＤＥ４	ＭＯＶＥ　Ｘ，Ｊ ＲＭＯＶ　Ｘ，Ｊ ＊Ｊが停止条件となります。Ｊ＝０となる停止条件なし

＊ＭＯＶＥの場合はＪを略さないでＭＯＶＥ　Ｘ，Ｙ，０として下さい。

＊停止条件は、入力ポート２４～３１までのビットパターンＪによって指定される停止選択です。停止の選択は上位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

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

△△の表

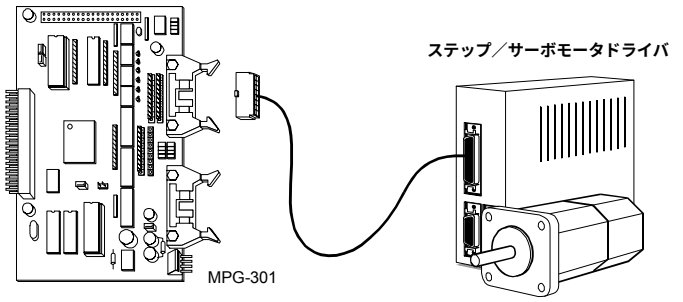
停止条件はビットによって指定されたポートの論理和となります。例えば、Ｊ＝３とします。この場合入力ポート２４、２５のどちらかがＯＮとなればパルス出力は停止されます。また、上位が０のため減速停止となります。尚、停止条件Ｊの使用方法はＲＭＯＶ，ＭＯＶＺ，ＲＭＶＺのいずれとも共通です。

３．８　ＭＰＧ－３０１の使用例

ＭＰＧ－３０１は汎用のパルス発生ＩＣ：Ｘ３２０２を搭載しており、次のような用途に対応できます。

①５０kppsを超える高速パルス発生
②パルス発生中での速度や加減即の変更、Ｓ字加減速などを必要とする用途
③エンコーダカウンタの入力

ＭＰＧ－３０１には別途、詳細マニュアルが用意されていますので、使用例についてはこちらを参照してください。このマニュアルでは簡単な例を一つ紹介します。接続は、ＭＰＧ－３０１のパルス出力をサーボモータあるいはステップモータのパルス入力に接続するだけです(ＭＰＧ－３０１の出力は差動出力ドライバです)。プログラム例ではパルスを連続発生させながら、入力０から７の値にしたがって速度を変更し、IN8をＯＮにすると停止するというものです。この例で見るようにＭＰＧ－３０１はリアルタイムで内部レジスタの参照・変更が可能です。



MPG-301 J4	ドライバ
1 (CW+)	Driver 入力　CW+
2 (CW-)	Driver 入力　CW-
3 (CCW+)	Driver 入力　CCW+
4 (CCW-)	Driver 入力　CCW-

※接続にはツウィストペアを推奨します。また、１００ｋ以上の高速パルス発生には、高速信号伝達に配慮されたケーブルをご使用ください。

```
"SW(8) ONで停止
'"N0-IN7 DSWの読み取り
ST_REG 0,250                "周波数倍率  250の時倍率は１／１
ST_REG 3,100                "起動周波数  100pps
ST_REG 4,1000               "最高周波数  1000pps
ST_REG 5,100                "加速ﾚｰﾄ     100pps²
ST_REG 6,100                "減速ﾚｰﾄ
CMND &HA0                   "動作完了ﾌﾗｸﾞﾘｾｯﾄ
CMND &H06                   "連続駆動
IF SW(8)=1 GOTO *STOP
R4=IN(0)
R4=R4*20                    "読み取り値の20倍の値を速度とする。
ST_REG 4,R4                 "周波数変更
GOTO *LOOP
*STOP
CMND &H31                   "減速停止命令
WAIT REG(-1)=&H20           "動作完了待ち
WAIT SW(8)=0
GOTO *MAIN
```

３．９　ＲＳ－２３２Ｃ通信

１）ＭＰＣ間の通信

ＭＰＣ間で通信をする場合は次の３つのコマンドで充分です。　参照：５章コマンドリファレンス
ＣＮＦＧ#，ＩＮＰＵＴ#，ＰＲＩＮＴ#

```
    MPC-1                          MPC-2

10 CNFG# 2 0 2                 10 CNFG# 2 0 2
20 ON 1                        20 INPUT# A
30 PRINT# 1                    30 ON 1
40 INPUT# A                    40 TIME 100
50 IF A=0 THEN #ERROR          50 IF SW(1)=0 THEN #ACK
60 OFF 1                       60 OFF 1
70 GOTO 20                     70 PRINT# 1
1000 *ERROR                    80 GOTO 20
1010   ・                      1000 *ACK
       ・                      1010 PRINT# 0
```

この例ではＭＰＣ－１がホストとなり３０で数値１を送出しています。これに対応するのはＭＰＣ－２のＩＮＰＵＴ#でＭＰＣ－１から与えられたメッセージを受け取ると次のステップ進みます。ＭＰＣ－１は、ＭＰＣ－２に指示を送出すると４０のＩＮＰＵＴ#でＭＰＣ－２からの信号を待ちます。ここで注意するのは、返ってくる数値には０と１の２種類があり、それによってＭＰＣ－１の継続作業も変更されます。このように通信によるインタロックでは、情報が数値として扱えるためインタロックを高度なものとすることができます。又、通信を受け取る側でパルス発生器として使用する場合は次の様になります。

```
    MPC-1                          MPC-2

      |                             5 *LOOP
100 PRINT# X1 Y1                   10 INPUT# A B
120 INPUT# A                       20 RMOV A B
130 IF A<>7 THEN *ERR              30 PRINT# 7
      |                            40 GOTO *LOOP
```

２）パソコンとＭＰＣの通信（ＣＨ１を使用した場合）

```
パソコン(Ｎ８８ＢＡＳＩＣ)      ＭＰＣ側

10 OPEN "COM1:E71XN" AS #1       5 CNFG# 2 0 2
           |                     10 INPUT A B
           |                     20 RMOV A B 0
100 PRINT#1,A,B                  30 PRINT# 7
110 INPUT#1,C                    40 GOTO 10
           |
           |
```

注意すべきことは、通信の手順をあわせることです。Ｎ８８ＢＡＳＩＣとＭＰＣの通信コマンドは良く似ているので簡単に通信が実現できます。

３）複雑なプロトコルに対応

ＴＮＹＦＳＣでは文字キャラクタの操作のために次の様なコマンドや関数がサポートされています。対パソコン、対ＭＰＣ等の通信の場合はこうしたコマンドを使用する必要は無いのですが、固定の通信フォーマットを備えた機器(計測器・表示器　等)に対しては、ＭＰＣ側でプロトコルを整合させる必要があります。

GET#()　PUT# PUTS#　GETN#　SKIP#　FIND#

ここでは、電圧計測器から次の様なフォーマットでデータが送られてくるものとします。

[SP]X1=230.13[SP][SP]Y1=156.17[SP][CR]

この場合、通常のＩＮＰＵＴ#文では”Ｘ１”や”＝”等の文字が邪魔になり、正常な数値の取り込みができません。こうした場合、次の様なプログラムを組めばデータを取り込むことができます。

```
100 SKIP# &H3D                   &Ｈ３Ｄは’＝’
110 GETN# X0                     ２３０を変数Ｘ０にとりこむ
120 SKIP# &H2E                   &Ｈ２Ｅは．’
```

```
130 GETN# X1                    １３を変数Ｘ１にとりこむ
140 SKIP# &H3D
150 GETN# Y0                    １５６を変数Ｙ０にとりこむ
160 SKIP# &H2E
170 GETN# Y1                    １７を変数Ｙ１にとりこむ
180 SKIP# &H0D                  ＆Ｈ０Ｄは ＣＲ’
190 X0=X0*100
200 X0=X0+X1                    Ｘ０には２３０１３がセットされる
210 Y0=Y0*100
220 Y0=Y0+Y1                    Ｙ０には１５６１７がセットされる
```

ＳＫＩＰ＃は相当する文字コードを検索し、ＧＥＮＴ＃は数字・文字列のみを変数に代入します。この例はピリオドを挟んで１つのデータを２つの整数データとして取り込み後で再合成しています。前記はデータの読み込みでしたが、装置によってはＲＳ－２３２Ｃでコマンドを与えるものもあります。デジボルに対して次の文字列を出力しなければならない時は例の様にします。

```
"CH1 DAT"[CR]
```

【例１】

```
100 'CH1 DAT
110 PUTS# STR(-1)
120 PUT# &H0D
```

【例２】

```
100 PUT# &H43,&H48,&H31
110 PUT# &H20,&H44,&H41
120 PUT# &H54,&H0D
```

このように文字を文字コードとして扱うことによってＴＮＹＦＳＣ内部でデータを処理できるようにしています。通信プログラムで他に有効なコマンドとして覚えておきたいのは、関数ＲＳ(ｎ)とＴＳＴ＃(０)です。ＲＳ(ｎ)はコマンドで処理されていないが、バッファにたまっている受信キャラクタの数を知らせます。ＴＳＴ＃(０)はＣＨ１だけに限られますがデータを読み出さずに受け取っているキャラクタを分類します。ＩＮＰＵＴ＃やＧＥＴ＃(０)はキャラクタを受け取っていなくてもキャラクタ待ちの状態となるため実行効率が悪くなります。ＲＳ(ｎ)やＴＳＴ＃(０)で受信状態を確認の上プログラムを進めるとより効率の良いプログラムを記述できます。

４）パソコンとＭＰＣの通信（ＣＨ０を使用する場合）

ここでは、プログラム用ＣＨをパソコンに接続して使用する方法について述べます。次の様な簡単な例に基づき専用プログラムをつくることにします。プログラムは、与えられたデータに基づいてポートをＯＮ／ＯＦＦします。

```
10 'SAMPLE
20 INPUT B
30 QUIT 1
40 OFF A
50 A=B
60 FORK 1,1000      ----------->
70 GOTO 20          これに対応する
1000 ON A           プログラムは次
1010 TIME 20        の様になります
1020 OFF A
1030 TIME 20
1040 GOTO 1000
```

```
10 'INITIAL
20 OPEN "COM1:E71XN" AS #1
30 PRINT#1,CHR$(&H1);
40 IF INPUT$(1,#1)=">" THEN #START
50 GOTO 40
60 '
70 *START
80 PRINT#1,"RUN";CHR$(&HD);
90 '
100 'MAIL LOOP
110 IF INPUT$(1,#1)="?" THEN *RSPNS ELSE 110
120 *RSPNS0
130 INPUT "PLEASE SET DATA PORT NUMBER ",A
140 IF A>B THEN *RESET
150 PRINT#1,A;CHR$(&HD);
160 GOTO 110
170 '
180 *RESET
190  INPUT "CAN I RESET FASIC !!(Y/N)",C$
200 IF C$="Y" THEN 30 ELSE *RSPNS
```

この例では、ＭＰＣのソフト・リセットが使用されています。０１Ｈコードで”ＰＲＩＮＴ＃１，ＣＨＲ$(&Ｈ１)；”によってＭＰＣはソフト・リセットします。ＢＡＳＩＣのプログラム中では、”＞”のレスポンスとＩＮＰＵＴの”?”をキー・キャラクタとして使用しています。エコー・バックは全て読み捨てとなります。こうしたやりとりで注意することは、ＭＰＣが割り込み禁止となっていないだろうか？ということです。割り込み禁止は、Ｚ版でのパルス発生時におこり、この時ＭＰＣはただ一つのキャラクタのみ受付可能でそれ以上は送らないようにします。また、レスポンス(＞)を待たずにキャラクタを送り続けるとプログラムの内容が破壊されることがあります。このように、ＣＨ０

に接続してＭＰＣを制御することもできますがこれにはＭＰＣのプロトコルついて詳細に知る必要があります。次に基本プロトコルについて解説します。ＭＰＣでは、次の様な基本プロトコルを持ちます。例外は、コマンド自身が通信出力を持つ場合、＜ＣＴＲＬ＞＋＜Ａ＞キーによるソフト・リセットの場合、さらにイニシャルのメッセージ出力等の場合です。例外コマンドは、ＴＥＡＣＨ，ＰＲＩＮＴ，ＩＮＰＵＴ等のデータ入出力コマンドに限られます。ここではまず、”ＯＮ　１”を例としてプロトコルを詳説します。

《ＴＥＲＭＩＮＡＬ》	《ＭＰＣ》	
0	0 [echo]	echo: ｴｺｰﾊﾞｯｸ
N	N [echo]	sp: ｽﾍﾟｰｽ
[sp]	[sp] [echo]	cr: ﾘﾀｰﾝ
1	1 [echo]	lf: ﾗｲﾝ･ﾌｨｰﾄを意味します。
[cr]	[cr] [echo]	
	命令実行	
	[cr]	
	[lf]	
	>	

このように命令をエコーバックし、実行終了とともに[１ｆ]＞を出力します。プログラムの入力も全く同様です。例えば、一行入力　”１０　ＯＮ　１”の場合は次の様になります。

《ＴＥＲＭＩＮＡＬ》	《ＭＰＣ》
1	1 [echo]
0	0 [echo]
[sp]	[sp] [echo]
0	0 [echo]
N	N [echo]
[sp]	[sp] [echo]
1	1 [echo]
[cr]	[cr] [echo]
	命令実行
	[cr]
	[lf]
	>

この資料でプロトコルの全てを前記の様に記述するのは困難でありますので、次の様な略記法を用います。エコー部は、(　)で囲まれた部分です。レスポンスは、キャラクタとヘキサで表記します。ヘキサは、[＆Ｈ＿]と表現します。

☆★☆★☆★☆　特殊プロトコル一覧　☆★☆★☆★☆

```
(NEW[&HD])
     [&HD][&HA]>
(10 ON 1[&HD])
     [&HD][&HA]>
(20 ON 2[&HD])
     [&HD][&HA]>
(30 TIME 100[&HD])
     [&HD][&HA]>
(LIST[&HD])
     [&HD][&HA]10[&HA]ON 1
     [&HD][&HA]20[&H9]ON 2
     [&HD][&HA]30[&H9]TIME 100
     [&HD][&HA]>
(INPUT A B[&HD])
     [&HD][&HA]?(1234 567[&HD])  [&HD][&HA]>
(PRINT A B[&HD])
     [&HD][&HA] 1234 567         [&HD][&HA]>
(TEACH[&HD])
     [&HD](X,Y,Z,U) 0 0 210 0 [XYZ,U] 100 100              (*Q)
     [&HD][&HA]>
```

実際の詳細が不明の場合はＦＴＭＷディスクに含まれるACTERM を使用して下さい。ACTERM は無手順ターミナルソフトで受け取ったコントロールコードも表示できます。

５）デジタル社タッチパネルとの接続

ＭＰＣ－８１６のＲＳ－２３２Ｃユーザーチャンネル(ＣＨ１)とデジタル社「プログラマブル表示器　ＧＰ７０シリーズ」のメモリリンク通信のサンプルです。メモリリンクではＭＰＣがホストとなり、ＧＰメモリの読込み／書込みでパネルをコントロールします。

例１)はＧＰの「「割り込み出力」を用いた例です。このプログラムを実行するには、ＧＰの画面をデザインする際に、部品の出力を「割り込み出力」として設定します。「割り込み出力」とは「ワードＳＷ」や「Ｔタグ」のワードアドレスをシステムデータエリア「１３」に割り当てると、それを押したときに「定数」をＲＳ－２３２Ｃ出力するというものです。各ワードＳＷやＴタグに固有の「定数」を設定しておけば、押されたものの識別が出来ます。ＭＰＣはその出力をＧＥＴ＃で待っているだけです。ランプ等のＭＰＣからの出力はＧＰの「W」コマンドで行っています。この方法の利点は、プログラムが簡単・常時ループしない・ＧＰのボタンを押したときのレスポンスが良い、などがあります。しかし、出力可能な定数は１バイト(＆ＨＦＦは不可)なので、これを超えるデータは得られません。ＭＰＣとリンクできる部品の種類と数も限られます。簡単なパネルデザイン向きです。
例２)はパネルからの読込みをＧＰの「Ｒ」コマンドで行っています。書込みは例１と同じＧＰの「W」コマンドです。この方法の利点はワードデータが扱えることです。ＧＰの「数値表示器」や「設定値表示器」などの１６ビットデータの読込みができます。欠点は、プログラムが複雑、常にループする、一度に多くのデータを入力するとレスポンスが低下するです。レスポンスについては、ページ単位で入力範囲を制限すれば良いでしょう(同一ページの部品のアドレスは１回のＲＥＡＤで読める範囲に設定する)。このサンプルでは１ループで４ワード(６４ビット)をＲＥＡＤしていますが、この程度ならレスポンスの低下はあまり感じられません。

☆ＧＰの通信設定

通信プロトコル： アスキー互換
通信設定： 伝送速度９６００ｂｐｓ、データ長８ｂｉｔ、ストップビット１、パリティ無
通信方式： ＲＳ－２３２Ｃ

※メモリリンクプロトコル、ＧＰ操作、ＧＰ作画についてはデジタル社の各取扱説明書をご覧下さい。

例１）

```
'************
' GP-H70
'  SAMPLE
'------------
' USE T-TAG
' & SYS Adr13
'============
'GPH70S10
'  980818
'************
 CNFG# 4,0,2
 FORK 1,*RS_RCV                  "入力用のタスク
*LP1
 A=600                           "書込みアドレス
 S=IN(0)                         "書込みデータ
 GOSUB *WRITE
 WAIT IN(0)<>S
 GOTO *LP1
*WRITE
 PUT# &H1B,&H57                  "ESC W
 S0=A^&HF000                     "ｱﾄﾞﾚｽ(hex表記)をｱｽｷｰｺｰﾄﾞで出力
 S0=S0/&H1000
 GOSUB *WRITE1
 S0=A^&H0F00
 S0=S0/&H0100
 GOSUB *WRITE1
 S0=A^&HF0
 S0=S0/&H10
 GOSUB *WRITE1
 S0=A^&H0F
 GOSUB *WRITE1
 S0=S^&HF000                     "ﾃﾞｰﾀをｱｽｷｰｺｰﾄﾞに変換して出力
 S0=S0/&H1000
 GOSUB *WRITE1
 S0=S^&H0F00
 S0=S0/&H0100
 GOSUB *WRITE1
 S0=S^&HF0
 S0=S0/&H10
 GOSUB *WRITE1
 S0=S^&H0F
 GOSUB *WRITE1
 PUT# &H0D
 RETURN
*WRITE1
 IF S0>9 GOSUB *A_F
```

```
 ELSE_GOSUB *0_9
 PUT# S0
 RETURN
*A_F
 S0=&H37+S0
 RETURN
*0_9
 S0=&H30+S0
 RETURN
 '************
*RS_RCV
 R=GET#(0)                      "入力待ち
 PRINT R
 OUT R,0
 GOTO *RS_RCV
```

例２）

```
'************
 ' GP-H70
 '  SAMPLE
 '------------
 'USE GP
 ' READ/WRITE
 '   COMMAND
 '============
 'GPH70S20
 '  980818
 '************
 S=0
 CNFG# 4,0,2
*LP1                            "READ,WRITEとも同一タスク
 A=600                          "書込みアドレス
 S=IN(0)                        "書込みデータ
 GOSUB *WRITE
 '
 FOR A=500 TO 503               "A=読込みアドレス
 GOSUB *READ
 NEXT A
 '
 GOTO *LP1
*WRITE
 PUT# &H1B,&H57
 S0=A^&HF000
 S0=S0/&H1000
 GOSUB *WRITE1
 S0=A^&H0F00
 S0=S0/&H0100
 GOSUB *WRITE1
 S0=A^&HF0
 S0=S0/&H10
 GOSUB *WRITE1
 S0=A^&H0F
 GOSUB *WRITE1
 S0=S^&HF000
 S0=S0/&H1000
 GOSUB *WRITE1
 S0=S^&H0F00
 S0=S0/&H0100
 GOSUB *WRITE1
 S0=S^&HF0
 S0=S0/&H10
 GOSUB *WRITE1
 S0=S^&H0F
 GOSUB *WRITE1
 PUT# &H0D
 RETURN
*WRITE1
 IF S0>9 GOSUB *A_F
 ELSE_GOSUB *0_9
 PUT# S0
 RETURN
*A_F
 S0=&H37+S0
 RETURN
*0_9
 S0=&H30+S0
 RETURN
 '************
```

```
*READ
 PUT# &H1B,&H52                  "ESC R
 S0=A^&HF000                     "ｱﾄﾞﾚｽ(hex表記)をｱｽｷｰｺｰﾄﾞで出力
 S0=S0/&H1000
 GOSUB *READ1
 S0=A^&H0F00
 S0=S0/&H0100
 GOSUB *READ1
 S0=A^&HF0
 S0=S0/&H10
 GOSUB *READ1
 S0=A^&H0F
 GOSUB *READ1
 PUT# &H30,&H30,&H30
 PUTS# 1                         "読込みデータ数＝１
 PUT# &H0D
 '
 SKIP# &H41                      "Aまで読み飛ばし
 INPUT# D                        "CRまで入力
 0=A-500
 OUT D,0
 RETURN
*READ1
 IF S0>9 GOSUB *A_F
 ELSE_GOSUB *0_9
 PUT# S0
 RETURN
```

６）ＭＢＫ－８１６の使用方法

前節ではタッチパネルインターフェースをＲＳ－２３２Ｃで実現しましたが、複雑な用途にはプログラムが煩雑になり保守しにくいものになります。本格的なタッチパネルの応用にはＭＢＫ－８１６が適しています。ここでは簡単にＭＢＫ－８１６の紹介をします。(詳しくは別途詳細マニュアルが用意されていますのでご利用ください。)ＭＢＫ－８１６を用いると次のようなメリットがあります。

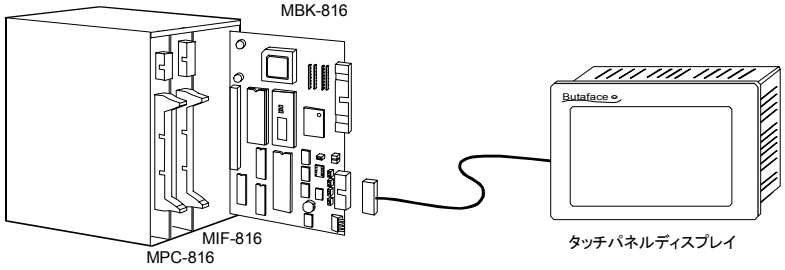
①ＲＳ－４２２接続となるため、信頼性が向上し、通信速度も速くなる。

②インターフェースコマンドがＯＮ／ＯＦＦ／ＳＷ()／ＯＵＴ／ＩＮ()などのＩＯコマンドになり扱いやすくなる

注意として次の事柄があります。

①ＭＢＫ－８１６の消費電流が大きいので設計時に注意

②IF SW(5000)=1 THEN ~という記述が出来ない。A0=5000, IF SW(A0)=1 THEN ~とする。



次は簡単なプログラム例です。実際にはＧＰ側をプログラムの上ＭＰＣのプログラムを設計しますが、これらのコマンド群は描画命令のため、ＭＰＣ側のＯＵＴコマンドだけで実現することができます。

```
OUT 3,44800                      "表示色
OUT 0,44900                      "背景色
OUT 1,45000                      "線種
OUT 110,45100                    "始点X
OUT 110,45200                    "始点Y
OUT 210,45300                    "終点X
OUT 200,45400                    "終点Y
ON 43205                         "四角形表示実行
WAIT SW(43205)=0                 "実行完了待ち
```

３．１０　いろいろなトラブル

１）トランジスタアレイの破損

ＭＰＣのトラブルで最も多いのがトランジスタアレイの破損です。これは通電したまま配線チェックしたり、配線チェックしないままパワーオンした場合に発生しています。いずれにしても初歩的なミスです。十分に注意して配線は確かめて下さい。トランジスタの破損はトランジスタアレイの交換で修理します。

２）バッテリバックアップ関係のトラブル

ＭＰＣ－８１６のＲＡＭは、リティウム電池でバックアップされています。これにより設定された変数、ポイントデータが無通電中も保持されます。しかし、バッテリバックアップはデリケートな保持方法のため相応の注意が必要となります。

☆組立時の注意

ＭＰＣ－８１６は無通電中にもリティウム電池によりＳＲＡＭが活かされています。このためＭＰＣ－８１６を鉄やアルミの板に直接置いたりアルミ箔に包んだりすると電池放電やＳＲＡＭのデータが破壊されます。又、静電気の放電を受けることによってもデータが変わってしまいます。組立時には前記を考慮の上扱うことと、組立後はＭＰＣＩＮＩＴの初期化を行うことが安全で確実です。

☆パソコンと接続した時

装置に組み込まれたＭＰＣと外部のパソコンを接続するときにＭＰＣのＲＳ－２３２Ｃポートに電気的なショックが発生します。特に、２００Ｖ系の装置ではこの度合が大きく甚だしい場合はインターフェースのＩＣを壊してしまいます。これは、パソコンと装置のアースを共通にすることにより回避できますが、現場ではなかなか難しいことです。ＭＰＣ－８１６では、ＲＳ－２３２Ｃインターフェース回路をフォトカプラで絶縁分離して外乱から内部回路を保護しています。

３）プログラム上の問題

ＭＰＣ－８１６に登載されたＴＮＹＦＳＣは１９８９年以来フィールドで実用に供されているソフトです。このため、バグは相当減らされていますが、仕様が古いためにいくつか概念上の誤解をまねいたり用法が不明確であったりする場合があります。

☆ＦＯＲＫについて

ＦＯＲＫはマルチタスクを発生させるコマンドですが、ふだん普通のプログラム(パソコン等)ではお目にかかれません。このため、性質がつかみにくかったり、初歩的な用法の誤りを招くことがあります。例えば、次の例です。

```
5 *LOOP
10  IF SW(1)=0 THEN *NEXT
20  FORK 1 *SUB
30  *NEXT
40  GOTO *LOOP
50  *SUB
60  ON 0
70  TIME 10
80  OFF 0
90  END
```

前記の例ではＳＷ(１)の値が１になった時にプログラム＊ＳＵＢが動作することを期待しているのですが、これでは＊ＳＵＢプログラムが動作しません。これはＦＯＲＫが実行された時点でプログラム＊ＳＵＢを実行し始めるという意味になるためです。この例では、ＳＷ(１)が１になっている限り何度もＦＯＲＫが実行されその度に＊ＳＵＢから再実行し始めます。つまり、プログラムが進行する前に再起動かけられてしまいます。正しくは次の様にします。

```
5 *LOOP
10 IF SW(1)=0 THEN *NEXT
12 FORK 1 *SUB
14 WAIT SW(1)=0
16 QUIT 1
30 *NEXT
40 GOTO *LOOP
```

☆反応が遅いと思われる場合

多くの入力で制御を分類する場合に、ＩＦ文とＳＷ(ｎ)関数を組み合わせて何行も記述すると反応が遅くなる場合があります。

```
5 *LOOP
10 IF SW(1)=1 THEN *WORK1
20 IF SW(2)=1 THEN *WORK2
30 IF SW(3)=1 THEN *WORK3
        (中略)
200 IF SW(20)=1 THEN *WORK20
210 GOTO *LOOP
```

このようなプログラム例では、５msec×２０＝０．１秒以内の反応が期待できません。これはＳＷ(ｎ)関数にノイズ除去のための５msecのフィルタが入っているためです。このような場合は、ＩＮ(ｎ)で入力関数を減らした方が有利です。

```
5 *LOOP
10 A=IN(0)~&HFE
20 IF A=2 THEN *WORK1
30 IF A=4 THEN *WORK2
40 IF A=8 THEN *WORK3
      |
50 IF A=128 THEN *WORK7
60 A=IN(1)
      |
```

☆ＨＳＷ(ｎ)を使用した場合の注意

先の節でとりあげたＳＷ(ｎ)は５msecのタイマーが組み込まれています。これに対してノイズ対策のタイマーを持たないのがＨＳＷ(ｎ)等Ｈの付された関数です。このためこの関数を使えばプログラムがシングルタスクである限り高速反応するプログラムとなります。問題はこれをマルチタスクで使用すると逆効果となることです。

```
10 FORK 1 *PRG1
20 FORK 2 *PRG2
30 *LOOP
40 IF A=1 THEN *SUB1
50 IF B=1 THEN *SUB2
      |
100 GOTO *LOOP
200 *PRG1
210   A=HSW(1)
220   GOTO *PRG1
230 *PRG2
240   B=HSW(2)
250   GOTO *PRG2
```

このプログラムでは、＊ＰＲＧ１と＊ＰＲＧ２の中でＨＳＷ(１)とＨＳＷ(２)を使うことによって高速を期待していますが逆効果となります。結果としてはＳＷ(１)，ＳＷ(２)とした方が総合的には高速となります。これは、マルチタスクにおける時間資源の有効活用という難しい内容になります。とりあえずここでは、１つのタスクでＨＳＷ()等を使用する場合にのみ高速応答が期待できるということを御了解下さい。

☆エラー停止

ＴＮＹＦＳＣはエラー発生と同時にインタプリタを停止して文番号とメッセージを表示しますがこの文番号はメインタスクの文番号となっています。これに対してエラーメッセージは発生したトラブルを表現していますのでマルチタスク下では読み方に注意が必要となります。

```
>LIST
10 FORK 1,*TASK2
20 FOR I=1 TO 10
30 ON I
40 TIME 5
50 OFF I
60 NEXTI
70 GOTO 20
80 *TASK2
90 PG 3
100 RMOV 1000,0
```

```
110 GOTO 100
>RUN
#40
!!Out of Range
>
```

前記の例ではあたかも＃４０で”Ｏｕｔ　ｏｆ　Ｒａｎｇｅ”となってしまったように見えますが、本当の問題は９０のＰＧ３で発生しています。マルチタスクでエラー停止した時はこの後にMONコマンドを使って停止文番号を参照します。

```
>MON
TASK1# 40 TASK2# 90
!!Out of Range
>
```

このようにMONコマンドで各タスクの停止番号を参照しどれがメッセージに対応するかを調べます。

４）瞬間停電について

ＭＰＣ－８１６Ｋ，ＫＦには瞬間停電検出機能があります。ＤＣ２４Ｖが著しく低下した場合、ＭＰＣの赤いＬＥＤが０．１秒程度の間隔で点滅します。ＭＰＣ－８１６の赤いＬＥＤの点滅は、プログラムのランタイムエラーでも発生します。この場合は一秒程度の間隔でゆっくりとした点滅です。瞬間停電の原因としては、ＡＣラインの停電のほかに、ＤＣ２４Ｖへの過負荷が考えられます。装置が間歇的に停止するような場合、赤いＬＥＤの点滅を確認して下さい。

３．１１　言語の制限

ＭＰＣ－８１６のＴＮＹＦＳＣは簡易言語です。簡単な整数演算と制限された分岐制御文しかありません。ＴＮＹＦＳＣではインタプリタの構造を単純にすることによって高速性を保っています。

１）変数の制限

ＭＰＣ－８１６で使用できる変数はＡ～Ｚまでのアルファベットと数字を組み合わせたものに限られています。合計２８６個の変数です。

```
A A0 A1  ･･･  A8 A9
B B0 B1  ･･･  B8 B9
C C0 C1  ･･･  C8 C9
|  |  |
Y Y0 Y1  ･･･  Y8 Y9
Z Z0 Z1  ･･･  Z8 Z9
```

又演算は、3byte長整数のみで表現できる範囲は±８３８８６０８です。（Z版では±３２７６７の2byte長整数です。）

２）配列

配列要素として用意されているのは、ＡＲ()です。ＡＲ()はＡＲ(０)～ＡＲ(３１)の３２個のみです。又、P版ではM(n)を使用することができます。ＡＲ()はＳＦＴＬ、ＳＦＴＲコマンドにてデータをローテーションすることができます。点データとしてＸ()、Ｙ()、Ｕ()、Ｚ()４つが用意されており配列として使用することができます。要素は各１～３００です。Ｘ(０)、Ｙ(０)、Ｕ(０)、Ｚ(０)は特別な意味を持ち配列として使用することができません。

３）コメント・文字列の限界

ＭＰＣ－８１６でのコメント行はシングルクォートに続く１２文字以内です。許される文字は英数及び ＾ ’を除く記号に限られます。ラベルは１１文字以内です。ラベルの場合は＊(アスタリスク)を頭に付します。文字は英数及び ＾ ’と‘　’を除く記号に限ります。コメント及びラベル行は文字列出力に使用できます。例えば、ＲＳ－２３２ＣのＣＨ１に対して ”ＡＢＣ ”という文字列を出力するサブルーチンを作る場合は次の様にします。

```
        GOSUB *ABC
            |
*ABC
```

```
PRINT# STR(-1)
RETURN
```

関数STR(-1)は１行前の文字列(ラベルもしくはコメント)を指定します。複雑なコメントを必要とする時は、ソースファイルにのみ可能ですが "(ダブルクォート)に続く文字がプログラムロード時に切り捨てられるためコメントとして使用できます。但しこの方法は、ソースファイルのみに有効なのでＭＰＣ上で見ることはできませんし、ファイルロードすると消えてしまいます。

４）制御文

ＭＰＣ－８１６の制御文はＦＯＲ～ＮＥＸＴ、ＧＯＳＵＢ、ＧＯＴＯ、ＩＦ～ＴＨＥＮ、ＩＦ～ＧＯＳＵＢの５つです。ＧＯＳＵＢ　及びＦＯＲ～ＮＥＸＴではスタックを用いた制御を行っているので、ネストの深さと制御文中からの飛び出しに注意して下さい。ありがちなトラブルは次の場合でいずれも禁止事項です。又、ＧＯＴＯとＧＯＳＵＢを混同して使用している場合も多くありますので注意して下さい。

【例１】

```
10 ON A
20 OFF B
30 GOSUB *WORK1
40 WAIT SW(1)=1
50 OFF 1
100 *WORK1
110 ON 1,2,3
120 OFF 5,4
130 GOTO 40 <---ＲＥＴＵＲＮで戻らずＧＯＴＯで戻すとエラーとなる
```

【例２】

```
10 FOR A=1 TO 10
20 ON 4,5
30 IF SW(5)=1 THEN 70 ----.
40 TIME 10                |  このようにＩＦ文などでＦＯＲ
50 OFF 4,5                |  文からは飛び出せない。何度も
60 NEXT A                 |  飛び出すうちにエラーとなる。
70 ON 1,2 <---------------'
80 OFF 4,5
```

【例１の修正】

```
10 ON A
20 OFF B
30 GOSUB *WORK 1
40 WAIT SW(1)=1
50 OFF 1
     |
100 *WORK 1
110 ON 1,2,3
120 OFF 5,4
130 RETURN
```

【例２の修正】

```
10 A=1
15 *LOOP
20 ON 4,5
30 IF SW(5)=1 THEN 70
40 TIME 10
50 OFF 4,5
60 A=A+1
65 IF A<11 THEN *LOOP
70 ON 1,2
80 OFF 4,5
```

制御用のスタックは各タスクとも４８byteとなっています。これに対してＧＯＳＵＢでは、４byteＦＯＲ～ＮＥＸＴでは６byteスタックを消費します。このためＧＯＳＵＢのみでは１１、ＦＯＲ～ＮＥＸＴのみでは８までのネストが許されますが、混在する場合は次の式で表現されます。

48 ≧ (GOSUBの深さ)*4+(FOR/NEXTの深さ)*6

又ＩＦ文では、ＥＬＳＥを使用することができます。ＥＬＳＥはＩＦ文の直後に置きます。ＥＬＳＥにはＥＬＳＥ__ＴＨＥＮとＥＬＳＥ__ＧＯＳＵＢがあります。（Ｐ版のみＺにはありません）

【例３】

```
IF SW(0)=1 THEN *SUB1
ELSE_THEN *SUB2
*SUB1        作業１
               |
*SUB2        作業２
```

【例４】

```
IF SW(0)=1 GOSUB *SUB1
ELSE_GOSUB *SUB2
*SUB1        作業１
             RETURN
*SUB2        作業２
             RETURN
```

５）記述の制限

☆一行に対する制限

残念ながらＭＰＣ－８１６では次のＩＦ文が記述できません。

a. IF X(1)=100000 THEN 200

しかし、次のＩＦ文は受付られます。

```
b. IF X(1)=10000 THEN 200
```

これは、ＴＮＹＦＳＣの内部の問題なのですがプログラム文はメモリ上で１６byte以内となっています。この規則はＩＦ文において限界となっています。ａ．では１００００００が3byteとられるのに対してｂ．の１００００は2byteです。ｂ．では１６byte以内に収まるのに対してａ．では１６byteを超えてしまいます。こうした場合インタプリタは！！Ｔｏｏ　ｌｏｎｇというメッセージを出力します。ＰＲＩＮＴ文では次の記述ができず、最後の７００００が略されてしまいます。

```
10 PRINT SQR(50000)､60000､70000
LIST
10 PRINT SQR(50000)､60000
>
```

☆引き数に対する制限

各コマンドの引き数は３つまでとなっています。例えば、ＯＮ、ＯＦＦコマンドでは次の記述が可能です。

```
ON 1
ON 1,2
OFF 1,2,3
```

プログラム中に４つ目の入力をしても無視されます。又引き数は変数、定数もしくは関数です。式は記述できません。

```
ON A                    変数
OUT IN(0),0             関数
OUT &HAA,0              定数
```

関数の関数も処理することはできません。

```
>ON IN(IN(IN(0))
???                     エラーとなる
```

６）演算に対する制限

演算は２項演算のみです。また、式の長さは１６文字以内です。次の様な記述をすると問題をおこします。

```
>10 A1=C(1000)^&HFF          （１５文字）
>LIST
10 A1=C(1000)^&H00FF         カーソルアップリタンする（１７文字）
>LIST
10 A1=C(1000)^&H000F         FFがFとなってしまう
>
```

これは、最初の入力の時１５文字以内であったのがリストをとるとヘキサデータが４文字固定表現のため１７文字となってしまいます。ここでカーソルを上に移動すると式が１７文字となり最後の１文字が略され意味が変わってしまいます。これは、プログラムをセーブすると同様の問題となりますので注意して下さい。ヘキサ表現はできるだけ変数に置き換えておいた方が安全です。

```
H0=&H00FF
H1=&HFF00
A1=C(1000)^H0
A2=C(1000)^H1
```

演算は＋，－，＊，／，％，＾，｜，ｘの８種です。＋，－，＊，／は電卓と同じく加減乗除を意味します。他は次の通りです。

％	余りを算出する	（パーセント）
＾	論理積	（ハット）
｜	論理和	（パイプ）
ｘ	排他的論理和	（スモールエックス）

７）通信の制限

ＭＰＣ－８１６では、ＣＨ１というユーザー用のＲＳ－２３２Ｃを備えていますが文字列処理はあまり得意ではありません。また、９６００bpsフルボーレートではプログラムの組み方によって問題を起こすことがあります。

☆文字列の扱い

ＩＮＰＵＴ＃、ＰＲＩＮＴ＃は文字列を扱うことができません。ＭＰＣとのやりとりは数値データに限った方が見通しが良くなります。ただ、例外的に次の様な応用があります。

ＳＴＲ()関数の使用

ＰＲＩＮＴ＃、ＰＵＴＳ＃ではＳＴＲ()関数を使用することができます。これにより、コメント文に書かれた文字列を出力することができます。

ＦＩＮＤ＃、ＳＫＩＰ＃、ＧＥＴ＃()

送られてくる通信データに文字列が含まれてくることはままありますが、ＭＰＣはこれを無視することができます。また、ＧＥＴ＃()により１文字をアスキーコードとして扱うことができます。

☆通信時のトラブル

```
10 GETN# A
20 A1=GET#(0)
30 SKIP# &H000D
```

前記のプログラムは一見うまく動作しそうなのですが、システム的な問題により２０のＧＥＴ＃(０)で正確に文字が読み取れないことがあります。これはＧＥＴＮ＃が内部的に割り込み禁止を伴う演算をしているために発生します。つまり、文字列を受取りながらＧＥＴＮ＃は処理を開始しますが、割り込み禁止とするためこの間文字を受け取らなくなってしまいます。これをさけるには、次のプログラムとするのが完全です。関数ＲＳ(１)によって文字列が貯まるのを待ち、バッファに貯まった文字列を処理するとこうした問題は発生しません。

```
5 WAIT RS(1)>10                 文字がある程度貯まるのを待つ
10 GETN# A
20 A1=GET#(0)
30 SKIP# &H000D
```

この問題は、ＩＮＰＵＴ＃でも同様に発生します。只、ＩＮＰＵＴ＃を使用する場合は、ＣＲ、ＬＦでターミネートされてからただちに次の文字列を受け取るプロトコルが少ないために問題となることはあまりありませんが、同様の問題をかかえていることに留意して下さい。