第 4 章 制御プログラムの基礎

BL/1 では、それぞれの制御アクチュエータに対応した様々なコマンドが用意されていますが、一度に習得するのは大変な作業となります。ここでは、制御内容に合わせて、いくつかの代表的なコマンドを紹介します。これによって、おおまかなプログラムイメージを描いていただけるものと思います。

それぞれのコマンドの使用方法については、コマンド・リファレンスを参照してください。

なお、実際的な応用例とプログラム記述については、別冊にて、MPC-2000 チュートリアルがあります。また、その内容に沿ったトレーニングキット XY03 も用意されております。

チュートリアルでは、トレーニングキットの装置に基づいて、XY 制御、タッチパネルインターフェース、CUnet の操作、通信の基本的なプログラムを網羅しています。当社 WEB でも常時資料公開しております。

4-1 I/O 制御

ON/OFF

BL/1 での ON/OFF 制御は、ON/OFF コマンドによって行います。たとえば、0.1 秒での ON/OFF の繰り返しは、以下のとおりです。DO ～ LOOP は制御文で、LOOP に出会うと、DO の直後に戻るという意味です。

```
DO
    ON 1
    TIME 100
    OFF 1
    TIME 100
LOOP
```

センサ・入力論理検出

次にあるセンサを検出して、検出したら 1 回、ON/OFF するというプログラムにします。

```
DO
    WAIT SW(193)==1
    ON 1
    TIME 100
    OFF 1
    TIME 100
    WAIT SW(193)==0
LOOP
```

最初の WAIT SW(193)==1 は、入力ポート 193 に接続されたセンサがオンになるのを待つという意味です。WAIT SW(193)==0 は、センサがオフになるのを確認しています。これにより、変化に対してのみ ON/OFF　を繰り返すことになります。

条件・論理演算

I/O 制御では、複雑な論理演算をすることもあります。たとえば、先の例では、SW(193) の条件のみでしたが、SW(192) もオンであることを条件に加えるには、以下のようにします。

(SW(193)&SW(192)) は、SW(192) と SW(193) の値の AND 演算です。このため、両方の値が 1 でないと、(SW(193)&SW(192)) の値は、1 になりません。これにより、両方オンという条件になります。

```
DO
    WAIT (SW(193)&SW(192))==1
    ON 1
    TIME 100
    OFF 1
    TIME 100
    WAIT SW(193)==0
 LOOP
```

なお、SW 関数には別途反転した値を持つ @SW() 関数が用意されています。

```
WAIT (SW(193)&@SW(192))==1
```

この場合、@SW(192) が 1 は反転論理のため、オフで 1 となります。したがってこの例では、193 がオン、192 がオフという AND 条件で成立することになります。

こうした論理式は WAIT 文のほか、IF,WHILE 文で使用しますが、式の値がすべて 1 になったときに正論理としています。

従って、WAIT SW(192)==1 と　WAIT SW(192) は同じタイミング待ちとなります。比較演算子 == は、比較結果が等しいときに 1、等しくないときに 0 となる演算のため、SW(192)==1 では、SW の値が 1 となり、さらに比較演算で 1 に等しくなるとしているからです。

これにより、SW の複雑な論理条件も簡単に記述することができます。

```
IF SW(192)|SW(193)|SW(194)|flag THEN     192,193,194 のいずれかが ON もしくは、変数 flag が 1 になったら成立
IF (SW(192)&SW(193))|SW(194) THEN       192 と 193 が双方 ON もしくは、194 が ON
IF (SW(192)&SW(193))|@SW(194) THEN      192 と 193 が双方 ON もしくは、194 が OFF
```

タイムアウト処理

タイムアウト処理には、timer_ (ダウンカウント変数) を使います。timer_ に正数を与えると 0.1 秒ごとにデクリメントされ 0 で停止します。したがって以下のような記述をすることによりタイムアウトを含んだタイミング処理が可能です。

```
timer_=1000
WAIT (SW(192)==1) | timer_==0
IF timer_==0 THEN : GOTO *TMOUT : END_IF
```

なお、timer_ 変数を外部タスクから参照、変更するには、TIMER() 関数が有効です。

文字列処理

主なコマンド

string$	末尾に $ を付けると文字列変数
FORMAT、STR$、HEX$、CHR$	書式、DEC→文字列、HEX →文字列、CODE →文字
VAL、ASC、HEX	文字列→数値変換
STRCPY、PTR$	複写
SERCH、SERCH$	検索
ptr_	文字列ポインタ

【コマンド使用例】

1) 文字列変数、連結

```
A$="" : B$="" : C$=""          /* 文字列変数初期化
A$="2007 年 "                  /* 文字列代入
B$="11 月 15 日 "              /* 文字列代入
C$=A$+B$                       /* 文字列連結
PRINT C$                       /* 表示

* 結果
2007 年 11 月 15 日
```

2) DEC→文字列変換 (書式無し)

```
D=20071115                      /* 数値
FORMAT ""                       /* 文字列書式初期化
D$=STR$(D)                      /* 数値→文字列変換
PRINT D$                        /* 表示

* 結果
20071115
```

3) DEC→文字列変換 (書式付)

```
D=20071115                      /* 数値
FORMAT "0000 年 00 月 00 日 "   /* 文字列書式指定
D$=STR$(D)                      /* 数値→文字列変換
PRINT D$                        /* 表示

* 結果
2007 年 11 月 15 日
```

4) HEX →文字列変換 (書式付)

```
D=&H20071115                    /* 数値 (16 進数 )
FORMAT "0000/00/00"             /* 文字列書式指定
D$=HEX$(D)                      /* 16 進数値→文字列変換
PRINT D$                        /* 表示

* 結果
2007/11/15
```

5) 内蔵クロック読み取り例

```
FORMAT "0000 年 00 月 00 日 "   /* 文字列書式設定
DT$=HEX$(DATE(0))               /* 年月日文字列取得
FORMAT "00 時 00 分 00 秒 "     /* 文字列書式設定
TM$=HEX$(TIME(0))               /* 時分秒文字列取得
PRINT DT$ TM$

* 結果
2007 年 11 月 15 日 12 時 34 分 19 秒
```

6) CODE →文字変換

```
A$=CHR$(&H41)+CHR$(&H43)+CHR$(&H43)+CHR$(&H45)+CHR$(&H4C)
PR A$

* 結果
ACCEL
```

7) 文字列→ DEC 変換

```
A$="NOV15,2007"
A=VAL(A$)                       /* 最初の数字文字列を得る
PRINT A

* 結果
15
```

8) 文字列→ CODE 変換

```
A$="NOV15,2007"
A=ASC(A$)                       /* 先頭の文字のコードを得る
PRX A
```

```
* 結果
0000004E                        /* &H4E='N'
```

9）文字列→ HEX 変換

```
A$="E07F"                       /* 16 進数として読める文字列
A=HEX(A$)                       /* 数値へ変換
PRX A                           /* 16 進表示
PRINT A                         /* 10 進表示

* 結果
0000E07F
57471
```

10）文字列複写 (そのままコピー)

```
A$="NOV15,2007"
B$=A$                           /* A$ を B$ にコピー
PR B$

* 結果
NOV15,2007
```

11）文字列部分コピー

```
A$="NOV15,2007"
STRCPY A$ B$ 3          /* A$ の 3 文字以降を B$ にコピー (A$ の第一文字を 0 として数えます )
PR B$

* 結果
15,2007
```

12）ポインタを使った部分コピー

```
FORMAT ""                       /* 文字列書式設定クリア
TT$=HEX$(TIME(0))               /* 現在時分秒取得
ptr_=TT$                        /* 文字列の位置を取得
ptr_=ptr_+2                     /* ポインタを 2 つ進める
HH$=PTR$(2)                     /* ポインタの位置から 2 文字切り出して HH$ に入れる
ptr_=ptr_+2
MM$=PTR$(2)                     /* ポインタの位置から 2 文字切り出して MM$ に入れる
ptr_=ptr_+2
SS$=PTR$(2)                     /* ポインタの位置から 2 文字切り出して SS$ に入れる
CL$=HH$+":"+MM$+":"+SS$         /* 文字列連結
PR TT$ "->" CL$                 /* TT$: 元の文字列 CL$: 合成後の文字列

* 結果
00090835 -> 09:08:35
```

13）文字を検索して部分コピー

```
a$="DATA X=AB0.4 Y=CD45 TEMP=DE55" /* 元の文字列
SERCH a$ "X="                   /* a$ の中の "X=" を探す。結果はポインタ ptr_ に入る
b$=PTR$(5)                      /* ptr_ の位置から 5 文字を b$ にコピー
ptr_=SERCH$("Y=")               /* ptr_ の位置から "Y=" を探して結果を ptr_ に入れる
c$=PTR$(5)                      /* ptr_ の位置から 5 文字を c$ にコピー
ptr_=SERCH$("TEMP=")            /* ptr_ の位置から "TEMP=" を探して結果を ptr_ に入れる
d$=PTR$(4)                      /* ptr_ の位置から 4 文字を d$ にコピー
PRINT b$ c$ d$

* 結果
AB0.4 CD45 DE55
```

4-2 タッチパネル接続

MEWNET プロトコル

MPC-2000 では、各シリアル・ポートに MEWNET 対応のタッチパネル・ディスプレイを接続することができます。

MPC-2000 に付属のシリアルポートは、RS-232 のみですが、拡張シリアルボード MRS-MCOM では、RS-422 でも使用可能です。プロトコルは MEWNET のみです。MEWNET はパナソニック電工のＦＡ用メモリリンクプロトコルです。

MEWNET プロトコルでは、PLC の複雑な接点機能に対応した、非常に多くの手順が用意されておりますが、MPC-2000 では、メモリエリアを DT 属性 ,I/O エリアを R 属性に想定しており、この 2 つに関係したプロトコルのみに対応しています。それ以外の属性を含んだパネル・プログラムや、MEWNET 対応とされている他社パネルでも接続できない場合がありますので事前に接続確認をお願いします。
現在まで、接続の確認がとれているタッチパネルは以下のとおりです。

Panasonic 電工	GT シリーズ (AIGT0030,AIGT2032 など)
デジタル	GP-2000/3000/4000 シリーズ
三菱	GOT シリーズ (GOT-2000)
キーエンス	VT3 シリーズ　(VT3-Q5M,VT3-W4T)
SAMKOON	SA シリーズ (SA-3.5A)
ミスミ	GX7 シリーズ

MEWNET を起動するには、以下の一行をプログラム先頭部分に書き加えます。
このコマンドは一度実行されると、プログラムの実行状態にかかわらずタッチパネルとリンクされます。リンクされれば、データが共有され、通信を意識することなく、タッチパネルにデータの表示、あるいはタッチパネルからのデータの設定を行うことができます。

```
MEWNET 38400 1
```

第 1 引数 38400 はボーレートを意味します。反応速度の点から 38400 を推奨します。
次の引数は使用するシリアルポートの CH 番号です。(キャラクタフォーマットは 8bit ノンパリティです。)

タッチパネル通信には、タスクがひとつ割り当てられますが、シリアルの CH 番号で決定されますので注意してください。また、1 部のタッチパネルで、奇数パリティ固定のものがありますが、この場合はキャラクタフォーマットを指定する以下の定数を追加します。

```
B7O  bit7 奇数パリティ
B7E  bit7 偶数パリティ
B8O  bit8 奇数パリティ
B8E  bit8 偶数パリティ
```

以下は、3800bps bit7 奇数パリティの場合

```
MEWNET 38400 1　B7O
```

使用されるタスク番号は 32-CH 番号となります。
したがって CH1 を指定した場合 (MPC-2000,2100 に付属の CH1) はタスク 31 がタッチパネル通信に引き当てられます。よって、この場合タスク 31 をプログラム中で使用したり不用意に QUIT すると、タッチパネル通信が損なわれます。

メモリ配置

タッチパネルとのメモリ共有は、MPC 側では、MBK() という予約配列を用います。MBK はワード型の配列で、8192 個確保されています。うち 0 ～ 7835 までをワードデータとして使用します。7836 ～ 7899 までは、システムがタスクごとの実行プログラム番号を常時書き込んでいます。
MBK(0) ～ MBK(7899) のエリアはタッチパネルでは DT0 ～ DT7899 に対応します。

【使用例】　　IF MBK(100)==10 THEN
　　MBK(200)=1000

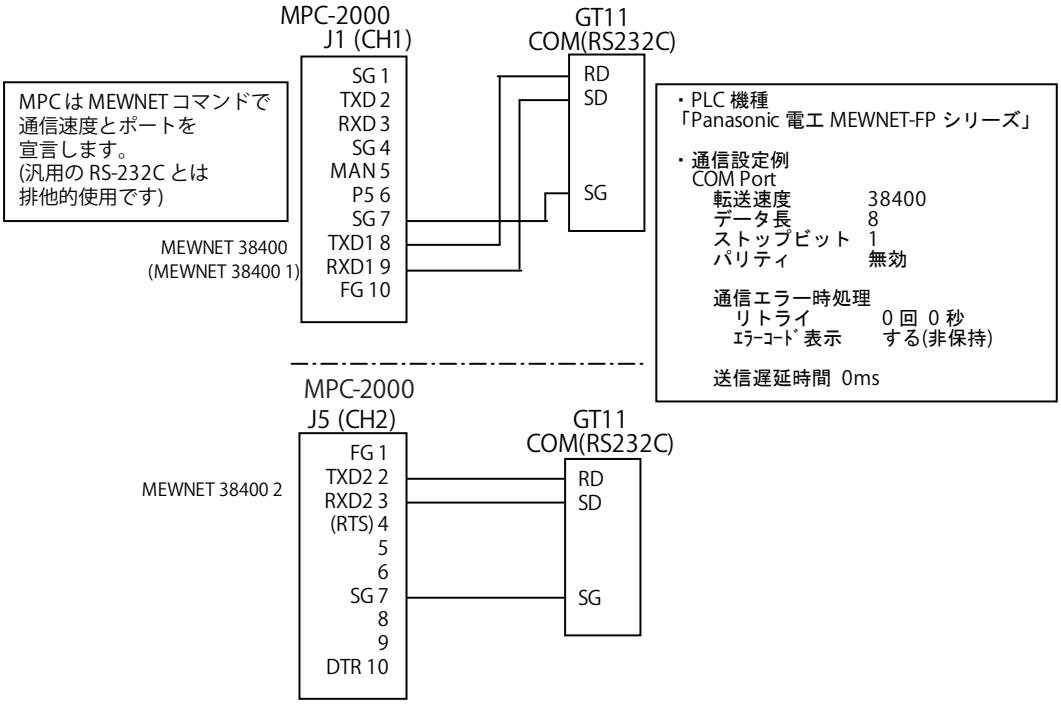
7900 ～ 7999 までは I/O エリアとしており、ON/OFF コマンドで操作することができます。
ON 70000 → バンク 0 のポート 0 をオン
OFF 70115 →　バンク 0 のポート 15 をオフ
このエリアはタッチパネルでは "R" で指定される I/O エリアとなります。

MBK エリア

0 ～ 7835	ワードデータ (DT に対応)	0 ～ 9 は一般的にシステムエリアとして使用される。 変更　S_MBK n m もしくは MBK(m)=n 参照　MBK(m)
7868 ～ 7899(Wrd) 7836 ～ 7899(Lng)	プログラム番号	リアルタイムでプログラム番号が更新されており、 タッチパネル側で参照、表示できる。 注)S_MBK LONG_PRG を実行すると Lng 型となる。
7900 ～ 7999	I/O エリア (R に対応)	0 ～ 100 バンク (YY) で 16bit ずつのエリア (XX) ON 7YYXX,OFF 7YYXX OUT 1000 7YY00~Lng SW(7YYXX),IN(7YY00)

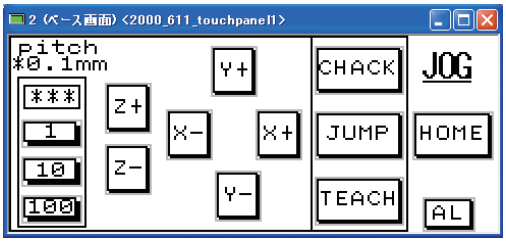
タッチパネル接続例

■ Panasonic 電工　GT11 との接続例

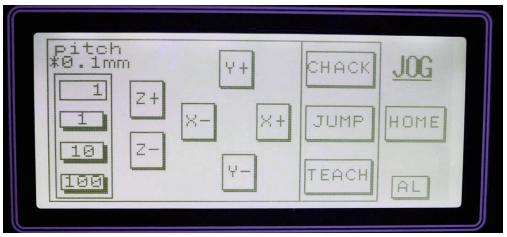


■ Panasonic 電工　GT11 での画面設計例

【イメージ図】



【実写図】



※写真は色を変えています。

【配置部品詳細】

表示	部品	基本設定	MPC コマンド例	
X+	スイッチ部品	モーメンタリ R200	SW(72000)	IN(72000~Wrd) (2 バイト読込)
X-	スイッチ部品	モーメンタリ R201	SW(72001)	
Y+	スイッチ部品	モーメンタリ R202	SW(72002)	
Y-	スイッチ部品	モーメンタリ R203	SW(72003)	
Z+	スイッチ部品	モーメンタリ R204	SW(72004)	
Z-	スイッチ部品	モーメンタリ R205	SW(72005)	
HOME	スイッチ部品	モーメンタリ R206	SW(72006)	
CHACK	スイッチ部品	モーメンタリ R207	SW(72007)	
JUMP	スイッチ部品	モーメンタリ R208	SW(72008)	
TEACH	スイッチ部品	モーメンタリ R209	SW(72009)	
AL	スイッチ部品	モーメンタリ R20A	SW(72010)	
1	機能スイッチ部品	値セット 出力先 DT101 値 1		
10	機能スイッチ部品	値セット 出力先 DT101 値 10		
100	機能スイッチ部品	値セット 出力先 DT101 値 100		
***	データ部品	参照デバイス DT101	MBK(101)	

【プログラム例】

```
MEWNET 38400                                    /* RS-232C CH1 を使う。

DO                                              /* SW 押下待ちのループ
    GT=IN(72000~Wrd)                            /* 2 バイト読み込み
    IF GT<>0 THEN : BREAK : END_IF              /* どれかが押されたらループを抜ける
    SWAP
LOOP

SELECT_CASE GT
    CASE &H01 : AX=X_A : MD=1  : GOSUB  *JOG_MV /* X+ SW
    CASE &H02 : AX=X_A : MD=-1 : GOSUB  *JOG_MV /* X- SW
    CASE &H04 : AX=Y_A : MD=1  : GOSUB  *JOG_MV /* Y+ SW
    CASE &H08 : AX=Y_A : MD=-1 : GOSUB  *JOG_MV /* Y- SW
    CASE &H10 : AX=Z_A : MD=1  : GOSUB  *JOG_MV /* Z+ SW
    CASE &H20 : AX=Z_A : MD=-1 : GOSUB  *JOG_MV /* Z- SW
    CASE &H40 : GOSUB  *JOG_HOME                /* HOME SW
    CASE &H80 : GOSUB  *JOG_CHACK               /* CHACK SW
    CASE &H100 : GOSUB  *JOG_JUMP               /* JUMP SW
    CASE &H200 : GOSUB  *JOG_TEACH              /* TEACH SW
    CASE &H400 : GOSUB  *ALIGN                  /* AL SW
    CASE_ELSE : PRINT "?"
END_SELECT
WAIT IN(72000~Wrd)==0
```

4-3 時間管理

MPC-1200,2000 及び 2200 では RTC が搭載され、日時時間を得ることができます。搭載 RTC は、エプソン・トヨコム製 RTC-7301 で月誤差 1 分程度となっています。MPC-1000,N816 にはこの機能はありません。

設定

カレンダ IC の使用にはまず初期設定が必要です。設定は SET_RTC コマンドを用います。

```
SET_RTC 2009 4 1          2009 年 4 月 1 日に設定します。
SET_RTC 12 2 0            12 時 2 分 0 秒に設定します。
```

設定した日時の確認は、date(0),time(0) 関数で行います。

```
#prx date(0)
20090401
#prx time(0)
00120204
#
```

時間検出

指定日時を検出するには、以下のように数値比較で行います。指定にはヘキサ型定数で指定します。

```
IF   TIME(0)==&H130500 THEN        (13 時 5 分 0 秒 )
IF   DATE(0)==&H20090401 THEN      (2009 年 4 月 1 日 )
```

以下の例では毎分 5 秒と 15 秒を検出しています。必要な桁のみを有効にすることによって毎時、毎日といった複雑な時間検出も可能です。

```
DO
      WAIT   &HFF&TIME(0)==&h0005
      PRINT   "time_05"
      WAIT   &HFF&TIME(0)==&h0015
      PRINT   "time_15"
LOOP
```

日時文字列

日時文字列は、DATE$(),TIME$() を用います。数値 0 ～ 2 によって様式が変わります。

```
10       FORMAT   "00000000"
20      a$=DATE$(0)
30       FORMAT   "000000"
40      a$=a$+TIME$(0)
60       PRINT   a$
#run

 2009090900141849
#pr time$(1)
 14:19:02
#pr date$(1)
  9/ 9/2009
#
```

4-4 軸制御

パルス発生ボードには MPC-1200 と MPG-2314 があります。MPG-2314 は、直線・円弧補間、センサ停止など複雑な用途に対応することができます。MPG-2314 は 10 枚まで拡張できるため、ソフト上では 10 枚× 4 軸に対応することができます。(ラックが最大 16 スロットのため、実際にはスロット数制限となります)
MPC-1200 は CPU ボード上に PGIC を備え、簡易的に低価格で軸制御に対応します。

PG のアサイン

どの PG を使用するのかは、PG コマンドで設定します。MPG-2314 が 0 ～ 9 の DSW 値に対応します。
MPC-1200 の PG は 17 にアサインされています。

加速度・速度

ACCEL、FEED、SPEED コマンドが用意されています。
ACCEL は最高速度と最低速度と加速度を決定します。S 字加減速の有無もここで指定します。FEED コマンドは 1 ～ 100(%) の引数を与えて最高速度の m% という速度指定を行います。対して SPEED コマンドは、pps 指定です。ACCEL で決定した、最高速度の範囲で m pps として速度を指定します。(ただし分解能は、最高速度の 1/8192pps となります。)

パルス発生コマンド

実際のパルス発生には、以下のようなコマンドがあります。用途により使い分けます。

コマンド	用途	説明
MOVS	位置決め	加減速度パルス発生　絶対位置指定　補間無し
RMVS	位置決め	加減速度パルス発生　相対位置指定　補間無し
MOVL	XY ステージ等	加減速度パルス発生　絶対位置指定　直線補間あり
RMVL	XY ステージ等	加減速度パルス発生　相対位置指定　直線補間あり
MOVT	塗布ロボット NC	軌跡制御連続パルス発生　絶対位置指定　円弧・直線補間
RMVT	塗布ロボット NC	軌跡制御連続パルス発生　相対位置指定　円弧・直線補間
RMVC	スピンドル等	無限パルス発生
STOP	汎用	パルス発生を途中で停止させるコマンド
HOME	補助コマンド	原点復帰マクロコマンド

設定とエラー

位置決めには、各種異常状態の検出や安全上必要なインターロックがあります。MPG-2314 にはリミット入力の外、サーボドライバエラー入力、検出停止入力があります。

コマンド / 関数	用途	説明
INCHK	保守	MPG 各種入力　表示
INSET	エラー入力設定	LMT 論理設定、ALM
STOP	停止条件入力	特殊原点復帰や途中停止条件を定める
PGE( )	停止原因一読取り	PG 停止後 :EMG,ALM,LMTn,LMTp|IN3,IN2,IN1,IN0
LMT( )	エラー要因読取り	常時参照 :EMG,ALM,LMTp,LMTn,SLMTp,SLMTn
HPT( )	IN0 ～ 3 入力	原点入力などの読み取り
RR( )	動作状態	PG 動作中かどうかの判定

初期設定　※以降のサンプルプグラムは MPG-2314 で作成したものです。

主なコマンド

PG	PG 選択
ACCEL、FEED	速度設定
INSET	入力設定

MPG-2314 を搭載しただけでは正常にパルスは出ません。初期設定が必要です。
最初に PG コマンドでタスクに MPG を引き当てます。次に ACCEL 等で初期設定をします。
ダイレクトコマンドでも可能ですが、最終的にプログラムに反映させてください。

【設定例】

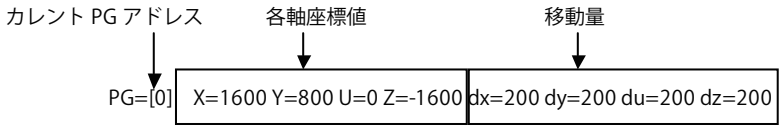
```
PG 0                          /* MPG-2314 ボード選択。MPG-2314 は DSW1 でアドレス設定。
ACCEL ALL_A 30000             /* 最高速・加減速設定。
FEED ALL_A 100                /* 使用速度設定 100%
INSET ALL_A ALM_ON|INP_OFF /* 入力機能設定。ｱﾗｰﾑは ON で有効、INPOS は OFF で有効とする。
CLRPOS                        /* 現在点を 0 クリア
```

ティーチモードでの動作確認

主なコマンド

PG	PG 選択
T(TEACH)	ティーチモード
PLS	点データー覧表示

パルス出力の最も簡単な確認方法はティーチモードです。FTMW 画面で T<Enter> でティーチングモードに入ります。



移動量 (1 回のパルス出力数) は 0~3 のキーで切り替えます。この値は SET コマンドで変更できます。
　　初期値 0：200 パルス／ 1：400 パルス ／ 2：600 パルス ／ 3：800 パルス

X,x,Y,y,U,u,Z,z キーで各軸が動作します。P キーでポイント番号入力。教示する点番号を入力して下さい。
Q キーでティーチングモードから抜けます。

最高速・加減速の設定

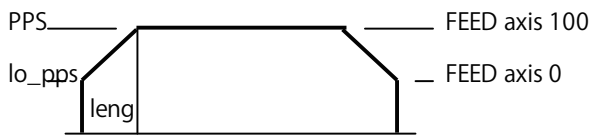
主なコマンド

ACCEL	最高速度・加減速・最低速度設定
FEED	速度指定

【書式】

```
ACCEL [axis] PPS [leng lo_pps]
  axis: 軸選択予約定数 ALL_A,X_A ～ Z_A
  PPS: 最高速度
  leng: 加減速領域パルス数
  lo_pps: 立ち上がりスピード ( 最低速度 )

FEED [axis] n
  [axis]: 軸指定予約定数 ALL_A,X_A ～ Z_A
  n: 速度指定 100( 最高速度 ) ～ 0( 最低速度 )
```

ACCEL と FEED の関係



MPG-2314 の入力チェック

MPG-2314 の入力ポートは INCHK コマンドで確認できます。

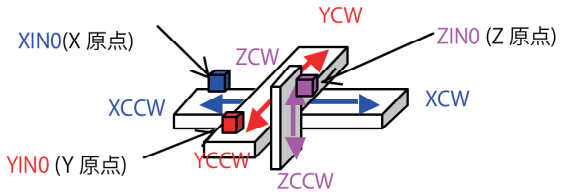
```
#PG 0                                                  /* PG 0 アサイン ( 後述 )
#INCHK                                                 /* MPG の入力確認
 MPG-2314
 X=+LMT:off-LMT:off ALM:off INP:off IN0:on  IN1:off    /* IN0= 原点 LS がオン
 Y=+LMT:off-LMT:off ALM:off INP:off IN0:on  IN1:off    /* IN0= 原点 LS がオン
 U=+LMT:off-LMT:off ALM:off INP:off IN0:off IN1:off
 Z=+LMT:off-LMT:off ALM:off INP:off IN0:off IN1:off
#                                                      /* どれかのキーでスキャン停止
```

原点復帰

主なコマンド

SHOM	原点入力設定
HOME	原点復帰動作
HPT	原点入力状態読み込み

XY03 の各軸には 1 個づつリミットスイッチが付いており、MPG-2314 の原点入力につながっています。



【サブルーチン例】

1）Z 単軸の原点復帰サブルーチン例

```
*Z_HOME
    PG 0
    ACCEL Z_A 10000 100 100  /* スピード設定。最高速 10KPPS、加減速領域 100 パルス、最低速度 100PPS
    IF HPT(ZIN0)<>0 THEN          /* XIN0 がオンなら退避移動
        RMVS Z_A -5000            /* CCW 方向に 1000 パルス移動
        WAIT RR(Z_A)==0           /* 動作完了待ち
    END_IF
    SHOM Z_A IN0_ON               /* 原点復帰設定。ZIN0 が ON するまで動け。
    TMOUT 10000                   /* 10 秒でタイムアウト
    HOME 0 0 0 50000              /* Z 軸 CW 方向に 50K パルス
    WAIT RR(Z_A)==0               /* 動作完了待ち
    IF Z(0)<>0 THEN               /* 動作後に座標が 0 でなければタイムアウト
        PRINT "Z TIME OUT"
    ELSE                          /* 動作後に座標が 0 なら HOME 完了
        PRINT "Z HOME"
    END_IF
    RETURN
```

2）X Yの 2 軸同時の原点復帰サブルーチン例

```
*XY_HOME
    PG 0
    ACCEL X_A|Y_A 10000 100 100          /* スピード
    FEED X_A|Y_A 100
    RMVL 5000 5000 0 0                   /* X,Y CW へ強制退避移動 (LS 確認を省略 )
    WAIT RR(X_A|Y_A)==0                  /* 動作完了待ち
    SHOM X_A|Y_A IN0_ON                  /* XY 軸それぞれ IN0 が ON になるまで動作
    TMOUT 10000                          /* 10 秒でタイムアウト
    HOME -100000 -100000 0 0             /* XY 軸同時動作
    WAIT RR(X_A|Y_A)==0                  /* 動作完了待ち
    RMVL 2000 2000 0 0                   /* 必要に応じてオフセット ( 電気的原点 )
    WAIT RR(X_A|Y_A)==0
    STPS X_A|Y_A 0                       /* X,Y 軸の現在位置を '0' にセット
    PRINT "XY HOME"
    RETURN
```

3）サブルーチンを呼び出すメインルーチン

```
GOSUB *Z_HOME          /* ハンドとワークの干渉を避けるため最初に Z 軸を原点復帰 ( 上昇 ) する
GOSUB *XY_HOME
END
```

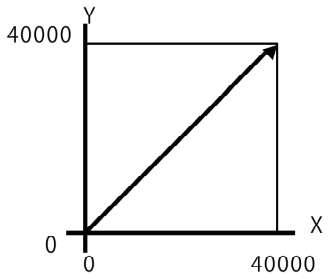
※これらのサブルーチンは後述のサンプルでも使用しています。

絶対座標移動

主なコマンド

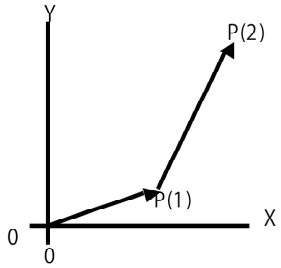
MOVL	直線補間移動
MOVS	単軸移動

①定数、変数で座標を指定して移動します。MOVL は直線補間します。



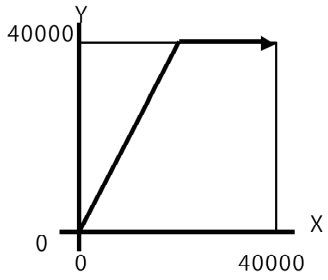

```
GOSUB *Z_HOME                    /* 前述の原点復帰サブルーチン
GOSUB *XY_HOME

ACCEL ALL_A 30000 3000 1000 /* 速度・加減速設定
FEED ALL_A 100                   /* 最高速度で動作
MOVL 40000 40000 VOID VOID   /* XY 軸絶対座標移動
WAIT RR(ALL_A)==0                /* 動作完了待ち

END
```

②ティーチングした点を指定して移動します。点はティーチングモードやプログラムで設定できます。
点番号を変数で指定することもできます。




```
GOSUB *Z_HOME
GOSUB *XY_HOME

ACCEL ALL_A 30000 3000 1000 /* 速度・加減速設定
FEED ALL_A 100                   /* 最高速度で動作
MOVL P(1)                        /* 点 P(1)へ直線補間移動
WAIT RR(ALL_A)==0                /* 動作完了待ち
PNO=2                            /* 変数指定
MOVL P(PNO)                      /* 点 P(2)へ直線補間移動
WAIT RR(ALL_A)==0

END
```

③ 到達点は①と同じですが MOVS は直線補間しません。ステップモータを使ったメカの振動防止、ステップ・サーボを組み合わせたメカで軸毎に異なるスピードを設定したい場合などに応用できます。



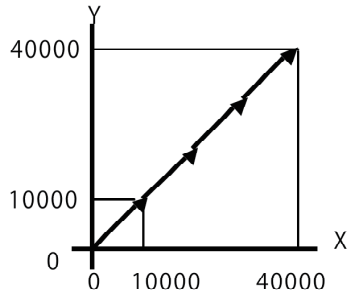

```
GOSUB *Z_HOME
GOSUB *XY_HOME

ACCEL X_A 15000 2000 1000     /* X 軸速度・加減速設定
ACCEL Y_A 30000 3000 1000     /* Y 軸速度・加減速設定
FEED ALL_A 100                /*  全軸最高速度で動作
MOVS 40000 40000 VOID VOID    /* X と Y は単軸動作
WAIT RR(ALL_A)==0

END
```

相対座標移動

主なコマンド

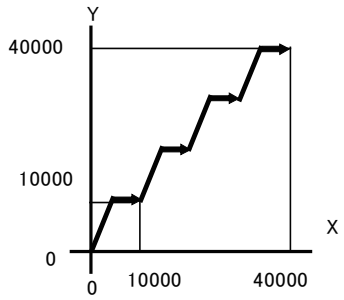
RMVL	直線補間移動
RMVS	単軸移動

①定数、変数で現在位置からの移動距離を指定して移動します。RMVL は直線補間します。




```
GOSUB *Z_HOME
GOSUB *XY_HOME

ACCEL ALL_A 30000 3000 1000 /*  速度・加減速設定
FEED ALL_A 100                /*  最高速度で動作

FOR I=1 TO 4                  /* 4 回繰り返し
  RMVL 10000 10000 0 0        /* XY 直線補間移動
  WAIT RR(ALL_A)==0
NEXT I

END
```

② 到達点は①と同じですが RMVS は直線補間しません。



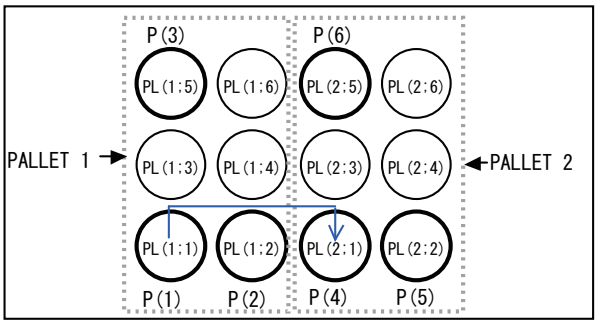

```
GOSUB *Z_HOME
GOSUB *XY_HOME

ACCEL X_A 15000 2000 1000     /* 速度・ 加減速設定
ACCEL Y_A 30000 3000 1000     /* 速度・ 加減速設定
FEED ALL_A 100                /* 最高速度で動作

FOR I=1 TO 4                  /* 4 回繰り返し
  RMVS 10000 10000 0 0        /* XY 単軸移動
  WAIT RR(ALL_A)==0
NEXT I

END
```

パレタイズ

主なコマンド

PALLET	パレット宣言
PL	作業点

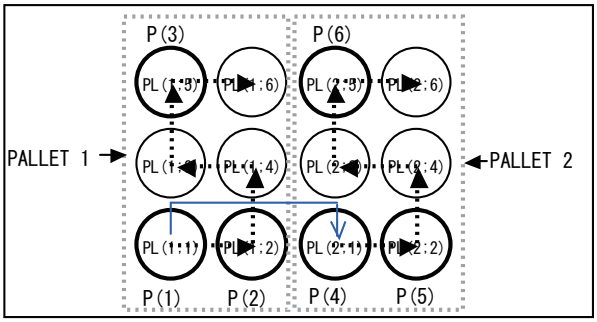
パレット間の移動に利用します。角の 3 点と行列数からパレット上の作業点 PL を算出します。




```
PALLET 1 P(1) P(2) P(3) 2 3          /* パレット宣言
PALLET 2 P(4) P(5) P(6) 2 3

GOSUB *Z_HOME
GOSUB *XY_HOME
ACCEL ALL_A 30000 3000 1000          /* 速度・加減速設定
FEED ALL_A 100                       /* 最高速度で動作

FOR M=1 TO 6                         /* PALLET 内の点 ※
      JUMP PL(1;M)                   /* PALLET 1 の点 M へジャンプ
      WAIT RR(ALL_A)==0
      ON 14                          /* チャック閉
      TIME 200
      JUMP PL(2;M)                   /* PALLET 2 の点 M へジャンプ
      WAIT RR(ALL_A)==0
      OFF 14                         /* チャック開
NEXT M

END
```

PL(n;m) の m が負の時、ZIGZAG モードになります。列間の移動距離が短くなります。

```
FOR M=-1 TO -6 STEP -1               /* 上記の※の行。負の引き数にする
```



※ PALLET に 4 つの点を指定すると歪んだパレットに対応できます。

途中停止

主なコマンド

STOP	パルス停止
INSET	MPG-2314 入力設定

■ ソフトによる途中停止
移動開始後に入力を監視してスイッチが入ったら STOP コマンドを発行しています。

```
GOSUB *Z_HOME
GOSUB *XY_HOME
ACCEL ALL_A 30000 3000 1000          /* 速度・加減速設定
FEED ALL_A 100                       /* 最高速度で動作

MOVL 40000 40000 VOID VOID           /* XY 直線補間
WAIT SW(194)==1                      /* 赤 SW オン待ち
STOP ALL_A STP_I                     /* 急停止。STP_D なら減速停止
WAIT RR(ALL_A)==0

END
```

■ ハードによる途中停止
下記は MPG-2314 のアラーム入力を利用した停止です。移動前に停止条件を設定しています。
移動中に X 軸アラーム (J6 コネクタ 13 番ピン) または Y 軸アラーム (同 14 番ピン) がオンになると両軸は即停止します。

```
GOSUB *Z_HOME
GOSUB *XY_HOME
ACCEL ALL_A 10000 3000 1000          /* 速度・加減速設定
FEED ALL_A 100                       /* 最高速度で動作

INSET X_A|Y_A ALM_ON                 /* アラーム入力設定

MOVL 40000 40000 VOID VOID           /* XY 直線補間
WAIT RR(ALL_A)==0

END
```

※ ALM などの MPG-2314 入力のチェックは INCHK コマンド。

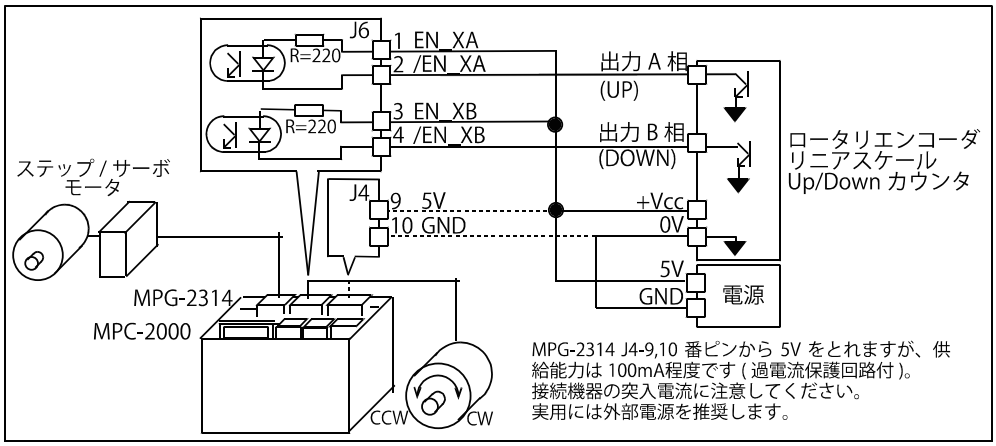
エンコーダ、カウンタ入力

MPG-2314 は標準で 2 軸のエンコーダ入力を備えています。2 相または Up/Down をコマンドで選択可能。
（オプションで 2 軸追加可能）

主なコマンド

INSET X_A PHASE1	2 相エンコーダ入力 1 逓倍
INSET X_A UP_DWN	Up/Down 入力に切り替える (デフォルトは 2 相)
INSET X_A CMP_CNT	COMP レジスタとカウント値を比較するモード
CLRPOS –1	X,Y,U,Z カウンタクリア
STPS X_C n	X カウンタを n に設定する
X(-1)	X カウンタの値を返す
X(-2,1)	X カウンタの値を返してカウンタをクリアする
CMP_C(X_A)	COMP レジスタと X カウンタの比較結果を返す

【接続例 (X カウンタ)】



ロータリエンコーダを回すと 100 カウント毎に出力を on/off します。

```
PG 0
INSET PHASE1                    /* 1 逓倍 1 回転で 1000 カウント
CLRPOS -1                       /* カンタクリア
OUT 0 0
DO
    NOW_XC=X(-1)                /* X カウンタ値読み込み
    IF (NOW_XC%100)==0 THEN     /* 100 で除算
      OUT @SW(0) 0              /* 出力反転 on/off
      PRINT NOW_XC SW(0)        /* 表示
      WAIT NOW_XC<>X(-1)
    END_IF
LOOP
```

・　実行結果

```
0 1
100 0
200 1
200 0
100 1
0 0
-100 1
```

(その他のサンプルプログラムは アプリケーションノート an2k-009 をご覧下さい)

MPC-1000 パルス発生機能について

MPC-1000 では、サブ CPU を 2 個搭載し、それぞれパルス発生器として使用することができます。

* この PG 機能は、コマンド通信に 0.1 ～ 0.2 秒程度要します。
* この PG 機能は、PIC の内部オシレータを使用するため、速度指定には +/-2% 程度の誤差があります。
* 加減速付きパルス発生のパルス幅は 15 μ秒固定です。

【占有ポート】

それぞれのパルス発生器は、PGA,PGB と名づけられており以下のポートを占有します。

	出力ポート	入力ポート
PGA	ON 12 (CW パルス) ON 13 (CCW パルス)	SW(192+12) READY SW(192+13) 通信用
PGB	ON 14 (CW パルス) ON 15 (CCW パルス)	SW(192+14) READY SW(192+15) 通信用

【PG の有効化】
PGA,PGB を有効にするには、それぞれ

```
ON PGA
ON PGB
```

を実行します。また、無効化するには、

```
OFF PGA
OFF PGB
```

とします。無効状態では、I/O を占有しなくなり、それぞれ制御用の I/O として使用できます。
また、ON/OFF は PG のソフトリセットも兼ねています。パルス発生を途中停止させる場合は、それぞれに対して、OFF PGA,OFF PGB を実行し 10msec 以上のオフ時間を確保してください。

【PG コマンド】PGX は PGA もしくは PGB

機能	コマンド	範囲	補記	READY
パルス方式	PGX “D” n	0 or 1	0: デフォルト 2PLS 1: 方向指示	
PWM	PGX “W” n	40 ～ 970	DA としても使用可	
PPS 指定パルス発生	PGX “G” pps	20 ～ 9000		
パルスレート設定	PGX “S” pps	20 ～ 9000		
パルス数指定パルス発生	PGX “P” count	-8000000 ～ 8000000		○
加減速テーブル生成	PGX “A” pps	500 ～ 12000		○
速度選択	PGX “F” n	10 ～ 0	n*10 %	
加減速パルス発生　相対	PGX “R” count	-8000000 ～ 8000000		○
加減速パルス発生　座標	PGX “M” count	-8000000 ～ 8000000		○
現在位置クリア	PGX “H” count		現在位置設定	
現在位置取得	PGX “C”		PGA は V_PGA	
バージョン取得	PGX “V”	20091105 以降	PGB は V_PGB	

*READY に○記号があるものは、指定数パルス発生など実行終了待ちが必要なコマンド

【使用方法 1】 パルス発生として

```
*PGAPGB
 TIME   300
 ON   PGA PGB                    PG の有効化
 WAIT   SW(192+12)==1            有効化確認
 PGA   "V" : PRINT   V_PGA       バージョン取得と表示
 DO
    FOR   i=20 TO 6020 STEP 1000
       PGA   "G" i                パルスレートを 20 ～ 6020 まで変化させる。
       TIME 100                   (CW)
    NEXT
    FOR   i=6020 TO 20 STEP -1000
       PGA   "G" 0-i              パルスレートを 6020 ～ 20 まで変化させる。
       TIME 100                   ( 負の値で CCW)
   NEXT
 TIME   100
 PGA   "G" 0                     G コマンド停止
 PGA   "S" 2000                  パルスレート設定
 PGA   "P" 1600                  1600 パルス発生　CW ( 加減速なし )
 WAIT   SW(192+12)
 PGA   "P" -1600                 パルス発生中は READY=0 となる。
 WAIT   SW(192+12)               1600 パルス発生　CCW ( 加減速なし )
```

【使用方法 2】 位置制御パルス発生として

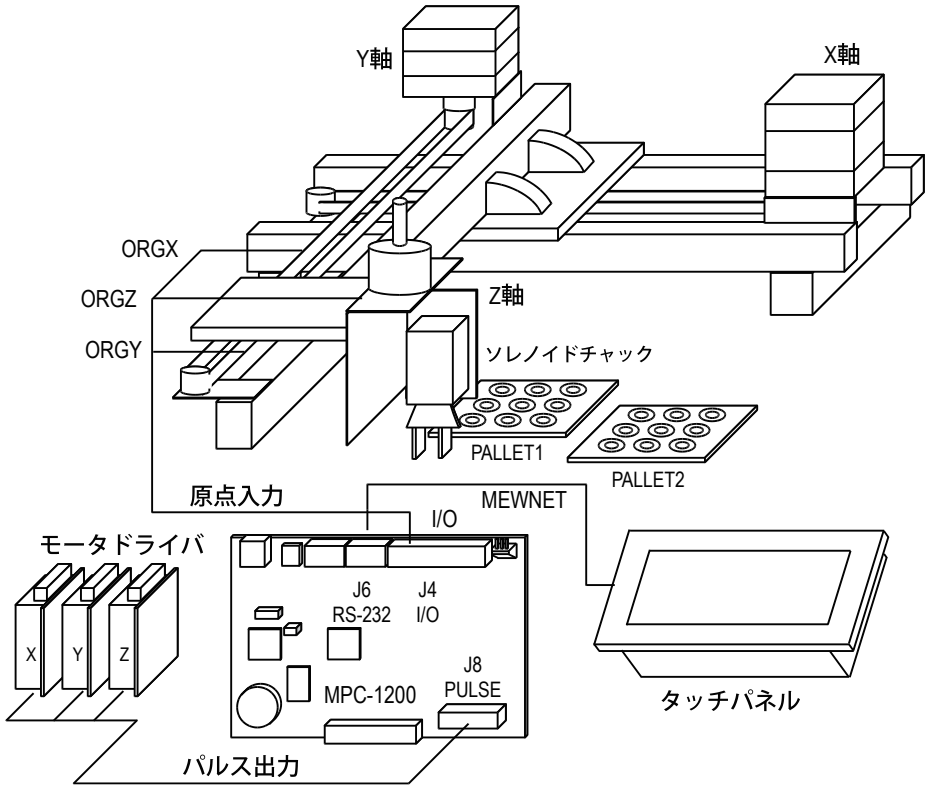
```
PGA   "A" 9000                    加減速度テーブルを 9000pps/s
WAIT   SW(192+12)                 テーブル生成終了待
PGA   "H" 0                       現在位置指定
'
FOR   j_=5 TO 10
   PGA   "F" j_                   速度指定
   FOR   i_=1 TO 10
      PGA   "R" 800               相対パルス発生
      WAIT   SW(192+12)
   NEXT
   PGA   "M" 0                    0 位置に移動。座標移動
NEXT
```

【コマンド解説】 PGX は PGA もしくは PGB の意味です。

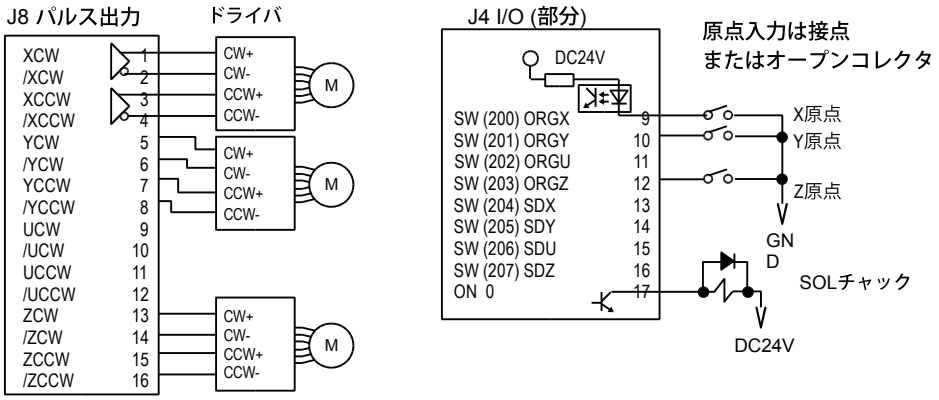
パルス方式	PGX “D” n	CW,CCW のニパルス方式か方向指示を指定します。 デフォルトは、ニパルス方式です。 方向指示に変更する場合は、PGX “D” 1 を実行します。
PWM	PGX “W” n	PWM パルス発生です。CW 側のみでパルス発生します。 デフォルトで 1kpps のパルスレートで n μ sec のオン時間を規定します。パルスレートを変更する場合は、PGX “S” pps を先に実行し、PGX “W” n をあとで実行します。 停止させるには、OFF PGA(PGB) するか PGX “W 0 を実行します。
PPS 指定パルス発生	PGX “G” pps	定速パルス発生です。指定したレートでパルスを発生します。 正の値で CW, 負の値をいれると CCW となります。 起動後、速度を変更することができます。 停止させるには、OFF PGA(PGB) するか PGX “G” 0 を実行します。
パルスレート設定	PGX “S” pps	PWM とパルス数指定パルス発生のパルスレートを決定します。
パルス数指定パルス発生	PGX “P” count	定速、指定パルス発生です。正の値で CW, 負の値で CCW 方向にパルス発生します。
加減速テーブル生成	PGX “A” pps	加減速付きパルス発生の速度テーブルを生成します。加速距離は、指定パルスレートの 1/10 で固定されています。 加減速は、フラッシュ ROM に固定されます。 フラッシュ ROM の書き換えは 10 万回未満とされていますので、注意してください。(同じ引数の場合は書き換えを行わないようになっています。)
速度選択	PGX “F” n	A コマンドで指定された速度を 10 段階に分割し、n/10 速度指定を行います。 加減速テーブルの変更は時間がかかりますが、F による速度変更は時間がかかりません。
加減速パルス発生相対	PGX “R” count	加減速付き、パルス発生です。座標管理されており、M コマンド、C コマンドと併用することができます。
加減速パルス発生座標	PGX “M” count	加減速付き、パルス発生です。現在位置と指定位置の差分パルス発生します。座標管理されており、R コマンド、C コマンドと併用することができます。
現在位置クリア	PGX “H” count	現在位置指定です。count を 0 にすると、原点となります。
現在位置取得	PGX “C”	結果は PGA,PGB それぞれ予約変数 V_PGA,V_PGB に返されます。
バージョン取得	PGX “V”	結果は PGA,PGB それぞれ予約変数 V_PGA,V_PGB に返されます。

MPC-1200 パルス発生機能について

MPC-1200 は 4 軸 PG-IC を搭載しており、シングルボードでも高速な多軸システムを構成できます。
下図は XYZ ロボットでパレット間のワーク搬送をするデモ機です。
パルスコマンドは MPG-2541 系に準拠しており、軸別制御、現在位置取得、S 字加減速等が可能です。
WARP コマンドにも対応しており高速 P&P を実現します。補間動作はできません。



パルス出力は差動出力 (26C31 相当) です。
原点入力は各軸 2 個有ります。原点として使用しないときは通常入力として使用できます。ドライバ側 Z 相などが差動出力の場合は I/F が必要です。



【パルスコマンド例】

```
PG 17                                /* PG アサイン MPC-1200 は PG 17
ACCEL X_A|Y_A 120000 12000 500       /* X,Y 軸加減速設定
FEED X_A|Y_A|Z_A 1000                /* X,Y,Z 最高速設定
SHOM &H55                            /* SD 入力無効化。この場合全 SD を無効にして通常入力使用
HOME X_A|Y_A NEG_L                   /* X,Y 軸原点復帰
RMVS X_A 1000                        /* X 軸相対座標移動
MOVS 50000 50000 0 0                 /* X,Y 軸絶対座標移動
LIMZ 20000                           /* Z 軸上昇制限
JUMP P(1)                            /* P(1) へゲートモーション
WARP 10000 PL(1;pln) 10000           /* Pallet1 の pln 番目に角のとれたゲートモーション
```

詳しくは次の資料をご参照ください。
アプリケーションノート「[an2k-047] 新型 XYZ デモ機 (XY-14)」
技術資料「MPC-1200 パルス出力例」

4-5 データ通信

RS-232/RS-485

MPC-2000 では 10CH のシリアル通信を扱うことができます。CPU ボード単体では、RS-232C しか扱うことはできませんが、MRS-MCOM を用いると、RS-422,RS-485 通信にも対応することができます。受信通信割り込みバッファは各 CH とも 256byte と十分に用意されおります。
MPC-1000 では CH1 を、MPC1200,MPC-2200 では CH2 を RS-485 として使用する事ができます。

1）設定
設定は CNFG# 1 "38400b8pns1NONE" というように CNFG コマンドを使用します。1200bps から 38400bps の各種フォーマットに対応します。RS-485 通信をする場合は、以下のようにします。

```
CNFG#  5 RS485 "38400b8pns1NONE"
```

予約定数 RS485 を引数に与えることによって、自動的に通信方向を切り替えるようになります。

2）送信
PRINT# コマンドを用います。PRINT# 文は、基本的には文字列を用いてください。変数も扱うことはできますが、書式を規定できません。文字列中に "¥n"(LF), "¥r"(CR), "¥t(TAB)", を用いることはできますが、その多の制御文字は、CHR$() を用います。

```
PRINT# CHR$(1) "DATA" CHR$(3)
```

3）受信
INPUT# コマンドを用います。INPUT# 文は文字列変数のみを引数に与えることができます。文字列として受け取ったあとに、VAL 関数 ,GET_VAL、SERCH コマンドなどによって内容を分析、データを取得します。

4）オプション
INPUT#,PRINT# 文には、受信文字数、タイムアウト時間、デリミタ、コードどを指定することもできます。また、特殊なオプションに COMPOWAY,STR_LEN があります。COMPOWAY は OMRON が規定する、ベーシック手順のプロトコルですが、このフォーマットの自動送信、受信をサポートしています。
SRT_LEN はヌルコードを含む文字列の送出に使用します。

RS-232C 機器との接続例

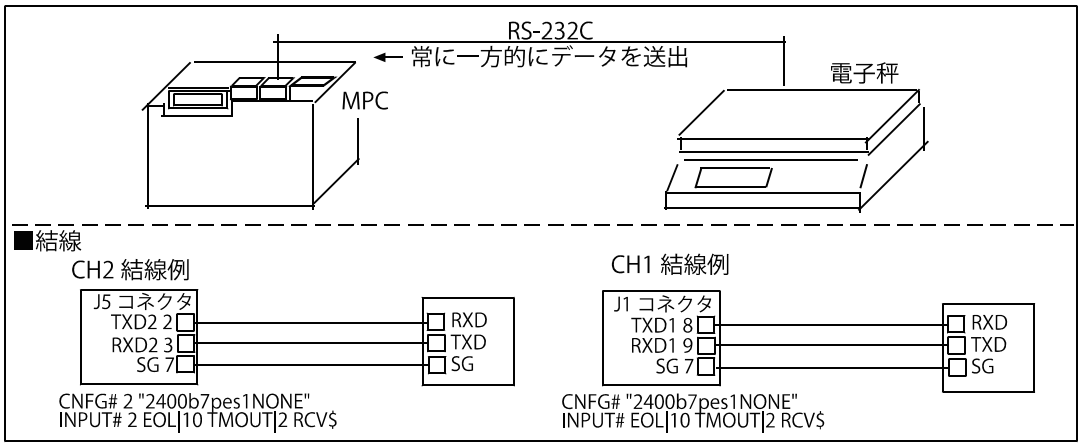
主なコマンド

CNFG#	通信設定
PRINT#	出力
INPUT#	入力

【受信文字列中から数値データを取り出している例】

```
/* このサンプルを実行するには CH1 の TXD1 と RXD1 を短絡してループバックします。
CNFG# 1 "9600b8pns1NONE"              /* 通信ポート初期化
FOR I=0 TO 20 STEP 2
    FORMAT "ABC0.0DEF¥n"              /* 文字列書式
    SND$=STR$(I)                      /* 送信文字列作成
    PRINT# 1 SND$                     /* 送信
    INPUT# 1 RCV$                     /* 受信
    PRINT RCV$ VAL(RCV$) VAL(0)       /* 受信文字列 最初の数値 次の数値
NEXT I

* 結果
ABC0.0DEF 0 0
ABC0.2DEF 0 2
ABC0.4DEF 0 4
```

【電子秤との続例】



■ 電子秤から送出される文字列データ

例) WT,+000000.0 g<CR><LF>
WT: ヘッダー文字。 WT= 安定時、US= 不安定時、OL= オーバー
+: 正負記号。負なら -。
000000.0： データ。 8 文字固定長、少数点は位置が変わったり、無かったりするかも ?。
g: 単位。
送出サイクル : 1 秒に 4 回弱垂れ流し。
MPC 側からの制御は無し

■ サンプルプログラム

```
CNFG# 2 "2400b7pes1NONE"              /* 初期化
FORMAT ""
TOTAL_CNT=0                           /* トータルカウント
RETRY_CNT=0                           /* リトライカウント
DO
*RETRY
  INPUT# 2 EOL|10 TMOUT|2 RCV$        /* LF まで受信 TMOUT 2 秒
```

```
    IF rse_<>0 THEN                 /* TMOUT 処理。rse_ は予約変数。必ず小文字。
      RETRY_CNT=RETRY_CNT+1
      PRINT "tmout retry" RETRY_CNT rse_
      GOTO *RETRY
    END_IF
  'PR RCV$
  ptr_=RCV$                         /* 文字列 RCV$ のポインタ。ptr_ は予約変数。必ず小文字。
  HEADER$=PTR$(2)                   /* RCV$ の先頭から 2 文字を HEADER$ へコピー
  ptr_=RCV$+14                      /* ポインタを 14 文字進める
  UNIT$=PTR$(1)                     /* ポインタ位置から 1 文字を UNIT$ へコピー
  SELECT_CASE HEADER$               /* ヘッダーをチェックする
    CASE "WT" : RESULT$=" ○ "
    CASE "US" : RESULT$=" △ "
    CASE "OL" : RESULT$=" × "
    CASE_ELSE                       /* 想定外
              PRINT "invalid header"
              GOTO *RETRY
  END_SELECT
  TOTAL_CNT=TOTAL_CNT+1
  WEIGHT1$=STR$(VAL(RCV$))          /* 文字列 RCV$ の中の最初の数値 ( この場合整数部 )
  SERCH RCV$ "."                    /* 秤の設定によっては小数部が無い場合がある ( 仮定 )
  IF ptr_<>0 THEN                   /* 検索文字が有った場合 = 小数部が有った場合。
    WEIGHT2$="."+STR$(VAL(0))       /* 文字列 RCV$ の中の次の数値 ( この場合小数部 )
  ELSE                              /* ↑ LCD 表示の都合で "." を "," に替えてます
    WEIGHT2$=""                     /* 小数部が無ければ空にする
  END_IF
  PRINT TOTAL_CNT RETRY_CNT RESULT$ WEIGHT1$ WEIGHT2$ UNIT$     /* FTMW 表示
  BUF$=HEADER$+WEIGHT1$+WEIGHT2$+"G    "      /* LCD には小英字は表示できない
  PR_LCD BUF$                       /* MPC-2100 LCD 表示 例 "WT117,3G"
 LOOP
```

実行結果 (FTMW 表示)

```
 1 0 ○ 0 .0 g
 2 0 ○ 0 .0 g
 3 0 ○ 0 .0 g
 4 0 ○ 0 .0 g
 5 0 ○ 0 .0 g
 6 0 ○ 0 .0 g
 7 0 ○ 0 .0 g
 8 0 ○ 0 .0 g
 9 0 △ 51 .3 g    ←秤に物を乗せる
 10 0 △ 111 .9 g
 11 0 △ 117 .0 g
 12 0 △ 117 .2 g
 13 0 △ 117 .3 g
 14 0 △ 117 .3 g
 15 0 △ 117 .3 g
 16 0 △ 117 .3 g
 17 0 ○ 117 .3 g
 18 0 ○ 117 .3 g
```

(see also: アプリケーションノート an2k-005)

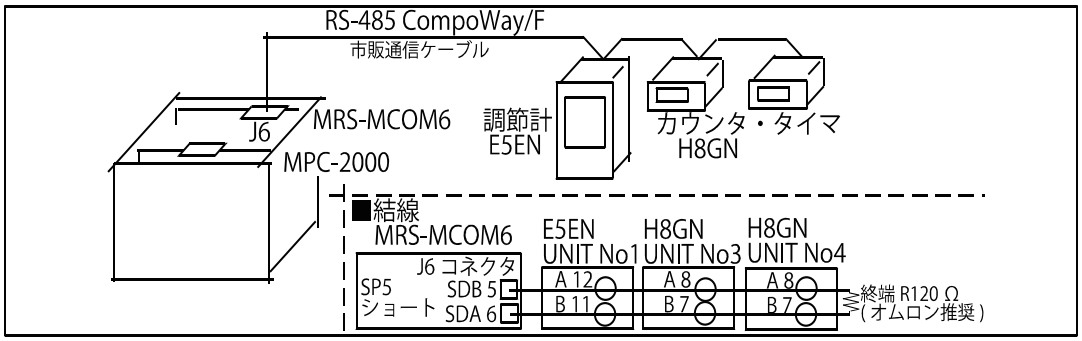
RS-485 機器との接続例

主なコマンド

CNFG#	通信設定 (パラメータに RS485 を指定します)
PRINT#	出力
INPUT#	入力
COMPOWAY	オムロン CompoWay/F プロトコルマクロコマンド / 予約定数

RS-485 は通信拡張ボード MRS-MCOM の J5,J6 コネクタでサポートされています。MRS-MCOM にはフェールセーフ回路が組み込まれているので機器側の終端抵抗以外の外付け回路は不要です。

【機器接続例】
オムロンデジタル調節計 E5EN、電子カウンタ / タイマ H8GN をマルチドロップ接続した例です。



※ CompoWay/F は、オムロン㈱の汎用シリアル通信における統一通信手順です。
※ RS-485 信号 A,B の呼称はメーカによって逆の場合があります。ご注意ください。
オムロンデジタル調節計 E5EN の変数エリアの現在値 (温度) を読み込みます。
CompoWay/F プロトコルのフォーマットに従って文字列を組立て、BCC を計算して送信、受信データから BCC を計算し必要な部分を切り出します。文字列処理は従来 (MPC-684) のスタイルです。

```
CNFG# 5 RS485 "9600b7pes2NONE"          /* MRS-MCOM CH5 RS485 設定
FORMAT ""                               /* 文字列フォーマット無し
SEND$=CHR$(2)                           /* STX
SEND$=SEND$+"01"                        /* ノード番号
SEND$=SEND$+"000"                       /* サブアドレス ,SID
SEND$=SEND$+"0101"                      /* MRC,SRC
SEND$=SEND$+"C0"                        /* 変数種別
SEND$=SEND$+"0000"                      /* 開始アドレス
SEND$=SEND$+"00"                        /* ビット位置
SEND$=SEND$+"0001"                      /* 要素数
SEND$=SEND$+CHR$(3)                     /* ETX
PUT_BCC=0                               /* 送信 BCC を計算
FOR I=1 TO LEN(SEND$)-1
      STRCPY SEND$ BUF$ I 1
      PUT_BCC=PUT_BCC^ASC(BUF$)&&HFF    /* 排他的論理和
NEXT I
PRINT# 5 SEND$ CHR$(PUT_BCC)            /* 送信
DO
  INPUT# 5 CHR_C|1 BUF$                 /* 1 文字ずつ受信
      IF ASC(BUF$)==&H02 THEN           /* STX( データの先頭 ) 待ち
          BREAK
      END_IF
LOOP
GET_STR$=""                             /* 受信文字変数 ( ﾚｽﾎﾟﾝｽﾌﾚｰﾑ STX 以降 ETX まで )
DO
  INPUT# 5 CHR_C|1 BUF$                 /* 1 文字ずつ受信
  GET_STR$=GET_STR$+BUF$
      IF ASC(BUF$)==&H03 THEN           /* ETX 受信なら LOOP 抜ける
          BREAK
      END_IF
LOOP
INPUT# 5 CHR_C|1 GET_BCC0$              /* 1 文字 (BCC データ ) 受信
GET_BCC0=ASC(GET_BCC0$)                 /* 受信した BCC データ -> 数値
GET_BCC1=0
FOR I=0 TO LEN(GET_STR$)-1              /* 受信文字列から BCC を計算
```

```
        STRCPY GET_STR$ BUF$ I 1
        GET_BCC1=GET_BCC1^ASC(BUF$)&&HFF
    NEXT I
    IF GET_BCC0<>GET_BCC1 THEN
        PRINT "BCC ERROR"
        PRINT " 受信 BCC=" HEX$(GET_BCC0) " 計算 BCC=" HEX$(GET_BCC1)
        END
    END_IF
    STRCPY GET_STR$ NODE$ 0 2                    /* 0 から 2 文字がノード No
    STRCPY GET_STR$ GET_TMP$ 14 8                /* 14 から 8 文字が温度
```

CompoWay/F 通信マクロコマンドを用いると、文字列の組立てが簡単に、BCC の計算が不要になります。

1）送信手順

* COMPOWAY コマンドで送信するテキストを構築します。
* PRINT# コマンドに COMPOWAY オプションを与えて実行すると STX と ETX、BCC を付加したコマンドフレームを送信します。

2）受信手順

* INPUT# コマンドに COMPOWAY オプションを与えて実行するとレスポンスフレームを受信し、BCC を計算します。
* COMPOWAY コマンドでレスポンスフレームから要素を変数に展開します。

3）COMPOWAY マクロコマンドでの通信例（文字列処理にはポインタを使っています）

```
CNFG# 5 RS485 "9600b7pes2NONE"               /* 通信初期化
FORMAT ""                                    /* 文字列フォーマット無し
/* コマンドフレームのテキスト部分の要素を変数・文字列変数に入れています。
node_no=1                                    /* ノード no
sub_adr=0                                    /* サブアドレス
sid=0                                        /* SID

mrc_src$="0101"                              /* MRC,SRC
hensu_shu$="C0"                              /* 変数種別
str_adr$="0000"                              /* 開始アドレス
bit_ichi$="00"                               /* bit 位置
yoso_su$="0001"                              /* 要素数
setteichi$=""                                /* 設定値 無し
cmnd_txt$=mrc_src$+hensu_shu$+str_adr$+bit_ichi$+yoso_su$+setteichi$  /* コマンドテキストを作成

COMPOWAY node_no sub_adr sid cmnd_txt$ snd$  /* ノード No からコマンドテキストまで結合して snd$ に入れます
PRINT# 5 COMPOWAY snd$                       /* コマンドフレームを送信します

INPUT# 5 COMPOWAY TMOUT|2 rcv$               /* レスポンスフレームを rcv$ に受信します
COMPOWAY rcv$ node_no sub_adr end_code res$  /* res$ にコマンドテキストの文字列が入ります

/* res$ の 0 から数えて 4 文字目から 4 文字がレスポンスコードです
ptr_=res$+4                                  /* ptr_ はポインタ予約変数。res$ の 4 文字目を指す
res_code=HEX(PTR$(4))                        /* ptr_ の位置から 4 文字コピー

/* res$ の 0 から数えて 8 文字目から 8 文字が読出データです
ptr_=res$+8                                  /* ポインタは res$ の 8 文字目を指す
res_data$=PTR$(8)                            /* ptr_ の位置から 8 文字コピー
PRINT res_code HEX(res_data$)                /* 温度表示

実行例
0 58                                         /* レスポンスコード =0、温度 58℃
```

(see also: アプリケーションノート an2k-004)

MPC-2000 シリーズでの USB メモリの運用について

■ 一般注意

1) USB メモリは、動作の確認されたメーカ品を使用してください。安売り品やノーブランド品には、読み書きにたえないものも、もともと信頼性の無い粗悪なものが存在します。

2) USB メモリは、消耗品としてお考えください。当社ベンチでも一週間程度、連続読み書きを繰り返しますと、USB メモリが破損するのを確認しています。この期間の長短は、USB メモリの品質に関係しています。このため、USB メモリはある程度使用しましたら、動作の確認されている新しいものと交換して使用してください。

3) USB メモリは、その動作には産業機器としての信頼性がありません。確実に動作することが必要な場合は、運用方法をよく吟味・試験してください。

■ 運用上の注意

1) MPC で使用する USB メモリは専用のものとし、パソコンで最初にフォーマットをしてから使用してください。

2) USB メモリで使用するファイル数は 30 個程度としてください。ファイル数が多くなると応答が遅くなる場合があり、タイムウアトなどのエラーが発生します。

3) MPC で使用できるファイル名はアスキー文字 "8 文字 +3 文字 " 形式のみです。使用する USB メモリにロング・フアイル名や日本語ファイル名のファイルをいれないでください。また、サプディレクトリは作成しないでください。障害の原因となります。

4) MPC では、FAT と FAT32 にのみ対応します。FAT12 は認識できません。

5) USB メモリはできるだけサイズの小さなものを使用してください (2G 以下推奨)。8G クラスの USB メモリでは、USB3.0 に対応していたり、セクタ数、セクタサイズが大きくなり、接続時の応答速度がかえって遅くなります。

■ パソコンとのデータ交換　(USB_PSAVE,USB_PLOAD)

MPC-2000 では USB メモリを使用した点データの読み取りおよび書き込み機能を備えています。この機能を利用すると、MPC の実行プログラムによって点データを書き出したり、読み込んだりすることができます。MPC プログラムの機種切り替えや、まとまったデータを PC に引き渡すのに有効です。

```
#USB_PSAVE P(1) 100 "PART1.P2K"
#NEWP
#USB_PLOAD "PART1.P2K"
#pr P(1) P(100)
 1 -1 1 -2 100 -100 100 -200
#
```

■ ログデータの書き出し (USB_WRITE)

稼動中に生産データなどをログファイルとして書き出すのに有効です。このコマンドでは、FILE$ で指定されたファイル名にデータを追加書き出しします。追加書き出しは、文字列を 128byte 単位にコマンド内で調整します。 改行コード (\n) を行末に付加すれば、文字列と改行の間に自動的にスペースを埋め込みます。(12_88 以後) 128 長にする理由は、USB メモリによっては、電源 ON/OFF 時にセクタをまたいだファイルの追記書き込みができないものがあるためです。128 文字長の文字列書き込みは、レコードのセクタ越えが発生しないため、不具合が生じません。

```
FILE$="LOG.TXT"
#FORMAT ""
#USB_WRITE "TIME=" HEX$(TIME(0)) "¥n"
#TYPE FILE$
TIME=00003856<- 全体で 128byte となるようにスペースを入れる –><CR><LF>
#
```

なお、128byte 化を抑止するには、AVOID オプションをコマンドに追加します。
#USB_WRITE　AVOID　"TIME=" HEX$(TIME(0)) "¥n"
* 電源 On/off を経たログ書き込みが無ければ、この方法でも問題ありません。

■ ファイルの読み出し (USB_READ)
　行単位でファイルの文字列を読み出します。ファイル名の指定は FILE$ で行います。下の例は、内容をすべて読み出し表示します。EOF(n) 関数は、ファイル読み取りが終端に達したかどうか判断するための関数です。値が 1 の場合ファイルの終端に達したことを示します。ファイルの読み取りを途中で停止する場合は、USB_READ -1 を実行します。

```
10　FILE$="AUTO.P2K"
20　 DO
30　  USB_READ　a$ : PRINT　EOF(0) a$
40　  IF　EOF(0)==1 THEN　: END　: END_IF
50　 LOOP
```

*USB_OPEN/PRINT/INPUT/USB_CLOSE を用いた書き出しは手順が煩雑で信頼性を損なうため非推奨です。

■ プログラムのセーブ・ロード (USB_LOAD,USB_SAVE)
　プログラムを USB メモリに保存、あるいは USB メモリから読み込むことが可能です。FTMW でロード可能なファイルであれば、使用できます。(12_88 以後)

■ RST-USB と USB(0) について

コマンド	機能	MRS-MCOM	MPC
RST_USB *	USB プロセスリセット	USB メモリの ON/OFF 無	USB メモリの ON/OFF 有
USB(0) **	USB の有無判定	無効 常に 1	USB メモリ有 :1 無し :0

*) RST_USB 後、MPC は USB メモリと初期化通信を開始します。このため、RST_USB 後、数秒間は、USB メモリに操作ができなくなります。MPC では USB() 関数の判定でこのタイミングを判断できます。

**) USB() 実行前に ON_USB を実行しておく必要がある場合があります。

■ USB メモリ関連のエラー

	番号	意味
USB_INUSE	53	ファイルはすでに使用中である
USB_NONE	54	USB メモリが接続されていない
USB_HALT	56	USB メモリが動作しなくなった
USB_NORSP	68	USB メモリプロセスが動作していない
NO_FILENAME	69	ファイル名が不適切
NO_FILE	70	指定のファイルが存在しない

* 動作エラー発生の場合 (56,68) は、RST_USB コマンドで USB メモリと USB メモリプロセスを初期状態に戻し、再度、同じ処理を繰り返す。

CUnet

MPC-2000 にはネットワーク機能も装備されており、これにより複雑なデータ通信を高速で行うことができます。使用ネットワーク CUnet は㈱ステップテクニカが開発し製造販売を行っている FA 用ネットワークで 512byte のメモリイメージをネット・ワーク上で共有することができます。ステーション数は最大 64 台まで、2.5msec 以内に共有メモリが同期するように設計されています。CUnet を使用するには、MPC-2000 側では、MPC-CUnet2、PC 側では、USB-CUnet が必要となります。

なお、MPC-2000 間のインターロック、簡単な数値情報の交換に限れば、IN,OUT,SW,ON,OFF などの IO コマンドにて共有メモリを直接、参照・変更することができるため、高速分散制御を容易に構築することができます。

また、MPC-2000 には CUnet のメール機能を利用した情報交換機能 (CU_POST,POST) があり、MPC ～ MPC 間、MPC ～ PC 間の点データ・MBK データエリアのブロック転送や文字列のやりとりを行うことができます。メールの転送単位は、P(n) が 15 個 (4byte*4*15)、MBK(n) が 120 個 (2byte*120) です。パソコン用の USB-Cunet 対応ソフトウェアは、専用 DLL にて VB 等のアプリケーションを容易に作成できます (デバイスドライバのセットアップが必要です)。

■ MPC-MPC 間の使用例

主なコマンド

CUNET	MPC-CUnet 初期化
SW,ON,OFF	ビット 2000 ～ 6095 操作
IN,OUT	バンク 2000 ～ 6095 操作
CU_POST	CUnet メールサーバータスク起動

以下のように 2 つの MPC を適切に初期化すると、お互いに CUnet 上のメモリを参照し、仮想的な I/O として使用することができます。

MPC-A 側	MPC-B 側
#cunet 0 8 31 #pr IN(SA_B(8)) 100 #on SA(0)+10	#cunet 8 8 31 #out 100 SA_B(8) #pr SW(SA(0)+10) 1 #

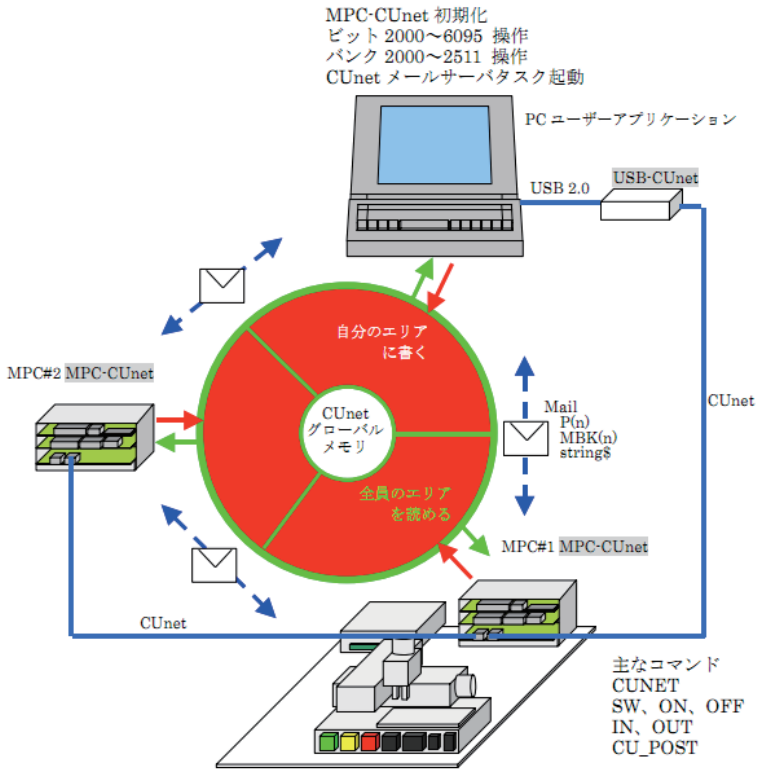
また、CU_POST,POST コマンドを使用すると、MPC 上の CU_POST サーバーが起動された MPC の点データを書き換えるえることができます。

以下の例では、B 側の POST コマンドによって、A 側に点データ (15 個分) が複写されます。

MPC-A 側	MPC-B 側
CU_POST #pls 1 P(1)　X= 46217　Y= 46218　U= 1　Z= 2 P(2)　X= 0　Y= 0　U= 0　Z= 0 P(3)　X= 0　Y= 0　U= 0　Z= 0 P(4)　X= 111　Y= 112　U= 0　Z= 0 P(5)　X= 104　Y= 105　U= 0　Z= 0 P(6)　X= 120　Y= 121　U= 0　Z= 0	POST 0 P(1)

■パソコンとの情報交換

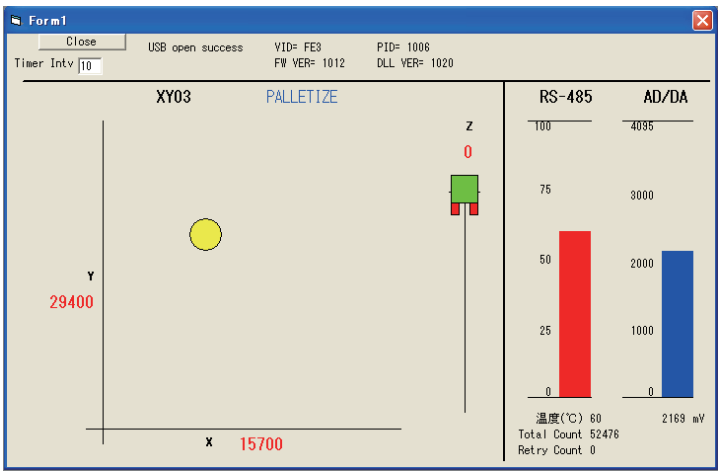
次の図は、MPC 二台とパソコンを連携させた概念図です。MPC～ MPC 間では高速インタロック、PC ～ MPC 間では、機種データのやりとりや操作情報の交換が可能になります。



■VB アプリケーション

・グローバルメモリ読み書き

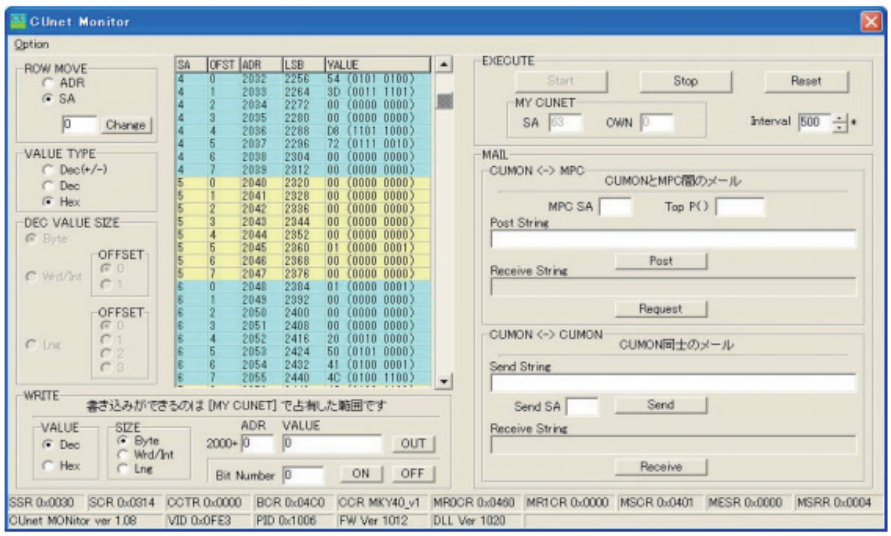
VB6 のサンプルです。MPC がグローバルメモリに書き込んだ XY03 座標値、RS-485 調節計 (温度)、AD 電圧等を読んで動作します。



■ モニタツール

こうしたネットワーク環境を構築するには、パソコンからデータの状態を参照したり、変更したりするツールが必要になります。

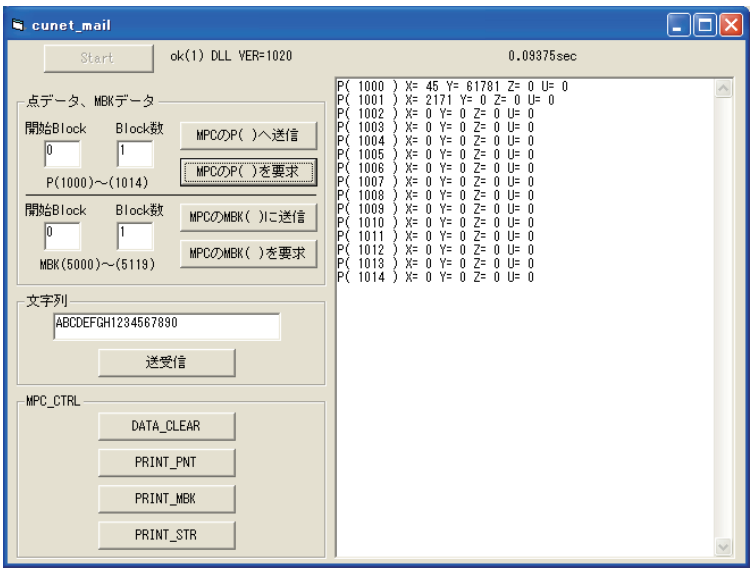
CUnet モニタ (CUMON.EXE) は、グローバルメモリの読み書きとメール送受信の確認を行うツールです。また、CUnet の本体チップである MKY40 のレジスタの状態も確認できます。セットアップ後やデバッグ時の動作確認に用います。弊社サイトから自由にダウンロードできます。



(see also: DOWNLOAD > TOOL > CUnetMonitor)

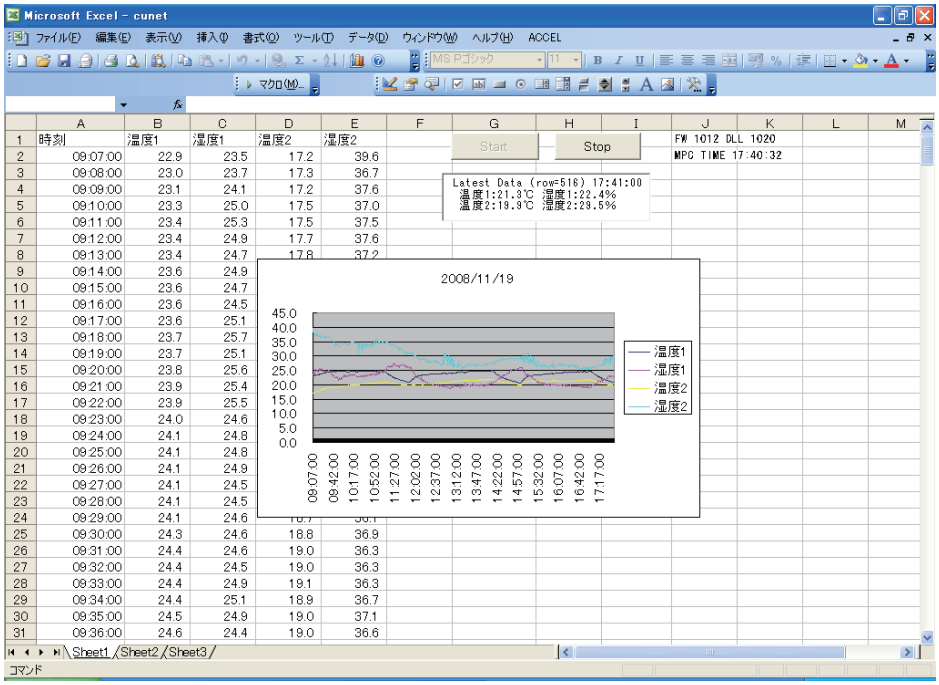
・CUnet Mail 通信

VB6 のサンプルです。MPC と点データ、MBK データのブロック転送や文字列のやりとりをします。



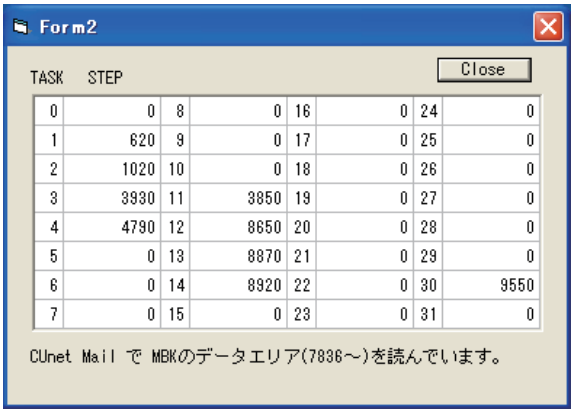
■ MS-EXCEL

MS-EXCEL VBA のサンプルです。下記は温湿度計を 2 台 RS-485 マルチドロップ接続し、測定値を一定間隔でワークシートに読み込み、同時にグラフを描画しています。



■ タスクモニタの例

USB-CUnet と MPC-CUnet を使い、前述の「タッチパネル接続」と同じように各タスクの実行分番号を知ることができます。VB6 サンプルアプリケーションでは CUnet Mail の cunet_reg_mbk 関数で、MPC の MBK エリアを読み込み MSFlexGrid に入れていきます。デバッグ、メンテナンスに利用できます。



0	0	8	0	16	0	24	0
1	620	9	0	17	0	25	0
2	1020	10	0	18	0	26	0
3	3930	11	3850	19	0	27	0
4	4790	12	8650	20	0	28	0
5	0	13	8870	21	0	29	0
6	0	14	8920	22	0	30	9550
7	0	15	0	23	0	31	0

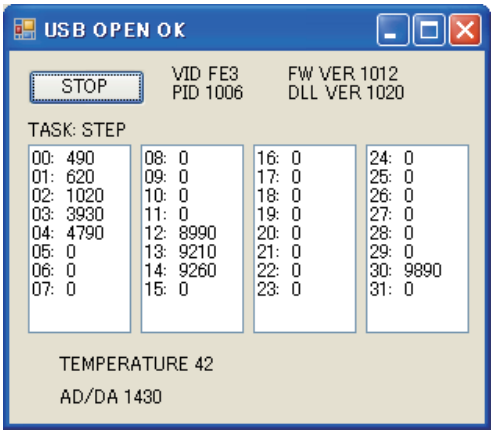
VB6 プログラム例 (Timer で定期的に読んでいます)

```
Private Sub Timer1_Timer()
Dim ar(0 To 119) As Long

  res = cunet_req_mbk(4, 7836, ar(0))
                    ' MBK ｴﾘｱ読込 (120 ﾜｰﾄﾞ ) ﾊﾟﾗﾒｰﾀパラメータ : 要求 SA, MBK() 先頭 , 格納配列
  i = 0
```

```
    For c = 1 To 8 Step 2
      For r = 0 To 7
        s = CStr(ar(i) + ar(i + 1) * &H10000)
                        ' MPC は「S_MBK LONG_PRG」指定してあるので 4byte 長にする。
        MSFlexGrid1.TextMatrix(r, c) = s
        i = i + 2
      Next r
    Next c

  End Sub
```

■ VB2008 Express Edition

VB2008 Express Edition での作成例です。CUnet-Mail でタスク文番号をモニタ、グローバルメモリ読み込みで温度と AD/DA 電圧を表示しています。



・VB2008 プログラム例 (Timer で定期的に読んでいます)

```
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Timer1.
    Tick
  Dim ar(0 To 119) As Integer
  Dim res, i, r, c As Integer
  Dim s As String

  res = cunet_req_mbk(4, 7836, ar(0))      ' MBK Area Read.  param= Request SA, MBK top addr, Storage
    array

  TextBox1.Clear()
  TextBox2.Clear()
  TextBox3.Clear()
  TextBox4.Clear()
  i = 0
  For c = 1 To 4
    For r = 0 To 7
      s = Format((i / 2), "00") + ":  " + CStr(ar(i) + ar(i + 1) * &H10000)
      If c = 1 Then TextBox1.SelectedText = s + Chr(13) + Chr(10)
      If c = 2 Then TextBox2.SelectedText = s + Chr(13) + Chr(10)
      If c = 3 Then TextBox3.SelectedText = s + Chr(13) + Chr(10)
      If c = 4 Then TextBox4.SelectedText = s + Chr(13) + Chr(10)
      i = i + 2
    Next r
  Next c

  Label5.Text = "TEMPERATURE " + CStr(cunet_in(2064, Cu_Int))    ' Global Memory Read
  Label6.Text = "AD/DA " + CStr(cunet_in(2080, Cu_Wrd))          ' Global Memory Read

End Sub
```

(このサンプルの全ソースは アプリケーションノート an2k-010 を参照してください)

4-6 アナログ制御

アナログ制御には、MPC-AD12 を用います。MPC-AD12 は、8CH の AD 入力、4CH の DA 出力に対応し、電源は、制御系と分離されています。

このため MPC-1200 等の簡易 AD 機能に比べて、計測精度・ノイズに対する安定性が期待できます。MPC-AD12 は二枚まで使用することができ、合計 AD 入力 16CH,DA 出力 8CH のアナログ制御に応用できます。

AD 変換

関数 AD( ) を用います。MPC-AD12 の標準状態では、0 ～ 4095 の値が得られますが、1digit が 1mv に対応します。A=AD(0) で A が 1000 となれば、入力が 1000mV つまり 1V であったということになります。AD 関数には、平均値取得モードもあり、MPC-AD12 によって自動的に平均値計算された値を得ることもできます。

なお、AD 変換 IC は IC ソケット上に実装された、アナログ・デバイス製 AD7890-4 を使用しています。この IC には、電圧のレンジの異なる、AD7890-10 というタイプがあり、こちらに換装することにより、+/-10V 対応となります。この場合、分解能は、10/2048mV=4.88mV/digit となります。AD7890-10 が必要な場合は、購入時に指定します。

また、MPC-AD12(CEP-125F 版) より、同期入力にも対応します。パルス列に対して、自動的にデータを取得する機能で、リアルタイム性を重視した AD 変換に対応できるようになります。

DA 変換

DA 出力にはコマンド DA を用います。DA 1000 1 と実行すれば、DA-CH1 に 1000mV つまり 1V が出力されます。

各種設定

AD コンバータの平均値のサンプル数設定や、AD7890-10 への交換時の設定など、SET_AD コマンドが用意されています。