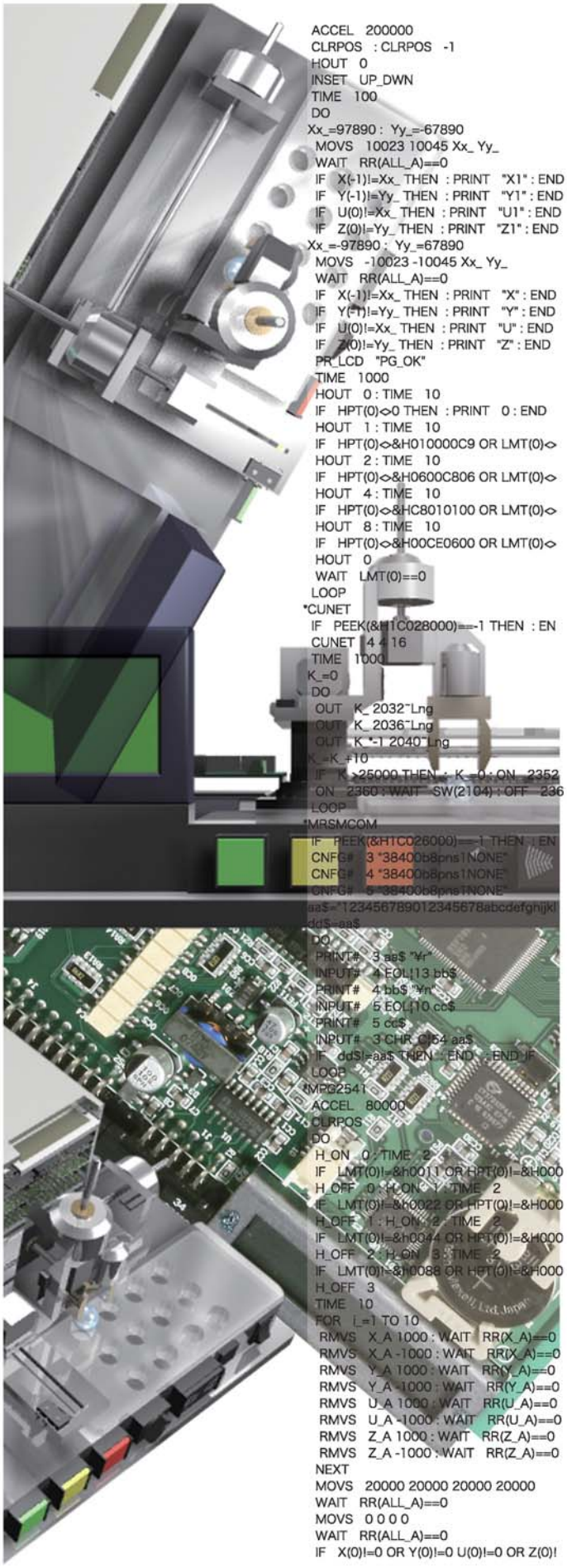
ACCEL
ACCEL Corporation

MPC-2000
Programing Tutorials



```
ACCEL   200000
CLRPOS  : CLRPOS  -1
HOUT  0
INSET  UP_DWN
TIME  100
DO
Xx_=97890 :  Yy_=-67890
 MOVS  10023 10045 Xx_ Yy_
 WAIT  RR(ALL_A)==0
 IF  X(-1)!=Xx_ THEN  : PRINT  "X1" : END
 IF  Y(-1)!=Yy_ THEN  : PRINT  "Y1" : END
 IF  U(0)!=Xx_ THEN  : PRINT  "U1" : END
 IF  Z(0)!=Yy_ THEN  : PRINT  "Z1" : END
Xx_=-97890 :  Yy_=67890
 MOVS  -10023 -10045 Xx_ Yy_
 WAIT  RR(ALL_A)==0
 IF  X(-1)!=Xx_ THEN  : PRINT  "X" : END
 IF  Y(-1)!=Yy_ THEN  : PRINT  "Y" : END
 IF  U(0)!=Xx_ THEN  : PRINT  "U" : END
 IF  Z(0)!=Yy_ THEN  : PRINT  "Z" : END
 PR_LCD  "PG_OK"
 TIME  1000
 HOUT  0 : TIME  10
 IF  HPT(0)<>0 THEN  : PRINT  0 : END
 HOUT  1 : TIME  10
 IF  HPT(0)<>&H010000C9 OR LMT(0)<>
 HOUT  2 : TIME  10
 IF  HPT(0)<>&H0600C806 OR LMT(0)<>
 HOUT  4 : TIME  10
 IF  HPT(0)<>&HC8010100 OR LMT(0)<>
 HOUT  8 : TIME  10
 IF  HPT(0)<>&H00CE0600 OR LMT(0)<>
 HOUT  0
 WAIT  LMT(0)==0
LOOP
*CUNET
 IF  PEEK(&H1C028000)==-1 THEN  : EN
 CUNET 14 4 16
 TIME  1000
 K_=0
 DO
  OUT  K_ 2032~Lng
  OUT  K_ 2036~Lng
  OUT  -K_-1 2040~Lng
  K_=K_+10
  IF  K_>25000 THEN  : K_=0 : ON  2352
  ON  2360 : WAIT  SW(2104) : OFF  236
 LOOP
*MRSMCOM
 IF  PEEK(&H1C026000)==-1 THEN  : EN
 CNFG#  3 "38400b8pns1NONE"
 CNFG#  4 "38400b8pns1NONE"
 CNFG#  5 "38400b8pns1NONE"
aa$="12345678901234567abcdefghijkl
dd$=aa$
 DO
  PRINT#  3 aa$ "¥r"
  INPUT#  4 EOL|13 bb$
  PRINT#  4 bb$ "¥n"
  INPUT#  5 EOL|10 cc$
  PRINT#  5 cc$
  INPUT#  3 CHR_C|54 aa$
  IF  dd$!=aa$ THEN  : END  : END IF
 LOOP
*MPG2541
 ACCEL  80000
 CLRPOS
 DO
  H_ON  0 : TIME  2
  IF  LMT(0)!=&h0011 OR HPT(0)!=&H000
  H_OFF  0 : H_ON  1 : TIME  2
  IF  LMT(0)!=&h0022 OR HPT(0)!=&H000
  H_OFF  1 : H_ON  2 : TIME  2
  IF  LMT(0)!=&h0044 OR HPT(0)!=&H000
  H_OFF  2 : H_ON  3 : TIME  2
  IF  LMT(0)!=&h0088 OR HPT(0)!=&H000
  H_OFF  3
  TIME  10
  FOR  i_=1 TO 10
   RMVS  X_A 1000 : WAIT  RR(X_A)==0
   RMVS  X_A -1000 : WAIT  RR(X_A)==0
   RMVS  Y_A 1000 : WAIT  RR(Y_A)==0
   RMVS  Y_A -1000 : WAIT  RR(Y_A)==0
   RMVS  U_A 1000 : WAIT  RR(U_A)==0
   RMVS  U_A -1000 : WAIT  RR(U_A)==0
   RMVS  Z_A 1000 : WAIT  RR(Z_A)==0
   RMVS  Z_A -1000 : WAIT  RR(Z_A)==0
  NEXT
  MOVS  20000 20000 20000 20000
  WAIT  RR(ALL_A)==0
  MOVS  0 0 0 0
  WAIT  RR(ALL_A)==0
  IF  X(0)!=0 OR Y(0)!=0 U(0)!=0 OR Z(0)!
```

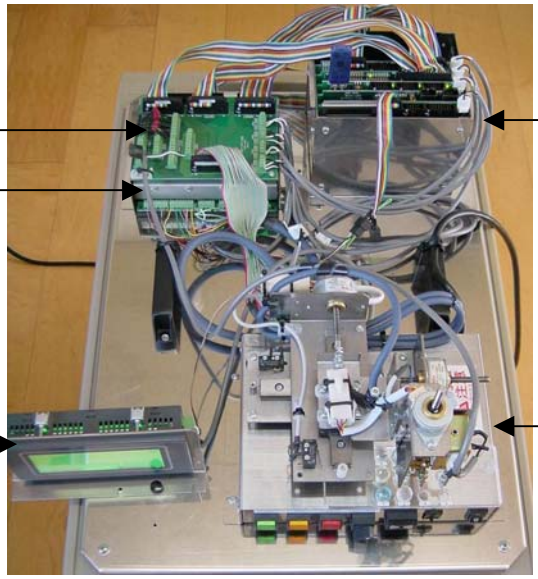
株式会社アクセル

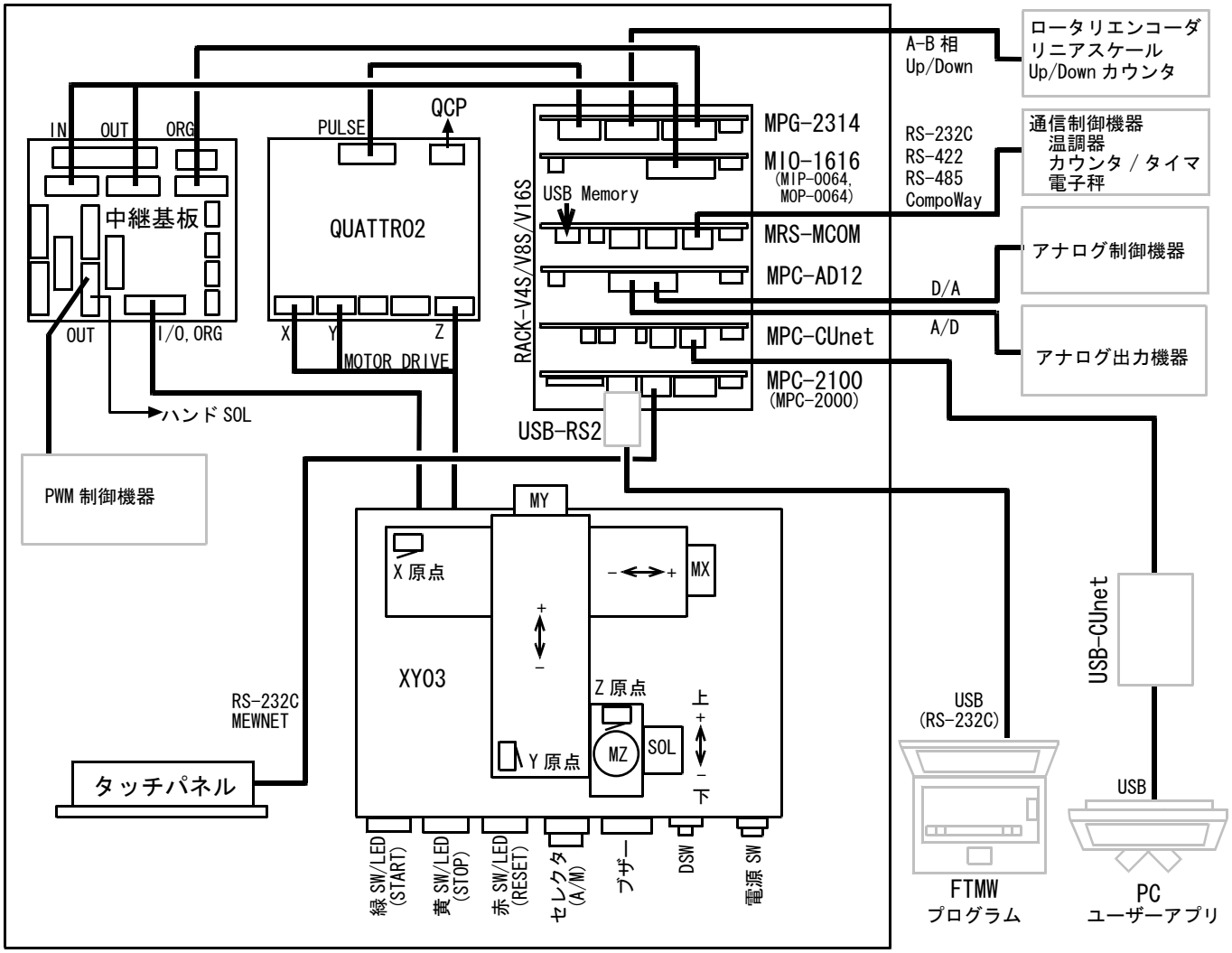
January 25, 2010

目次

トレーニングキット MPC-XY03 の構成と操作

◆ 構成例





※淡色表示の機器はトレーニングキット(貸し出し機)に含まれません。

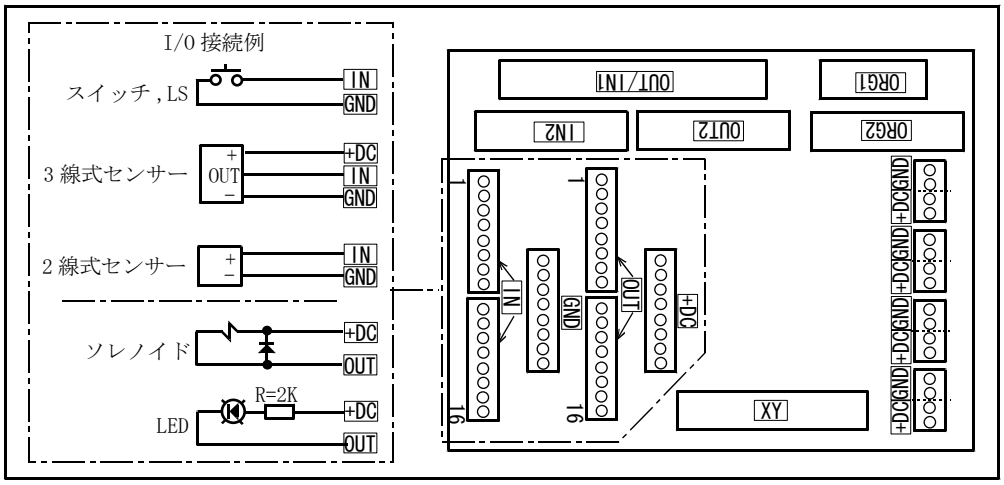
◆ XY03 I/O MAP

IN		OUT		MPG-2314 ORG	
ポート No	接続機器	ポート No	接続機器	機能	接続機器
192	緑 SW	0	緑 LED	XS1	X 原点 LS
193	黄 SW	1	黄 LED	YS1	Y 原点 LS
194	赤 SW	2	赤 LED	ZS1	Z 原点 LS
195	セレクタ SW	3	ブザー		
196~199	DSW	14	(SOL)		

緑・黄・赤 SW、各原点 LS は N.O.接点です。

◆ 中継基板

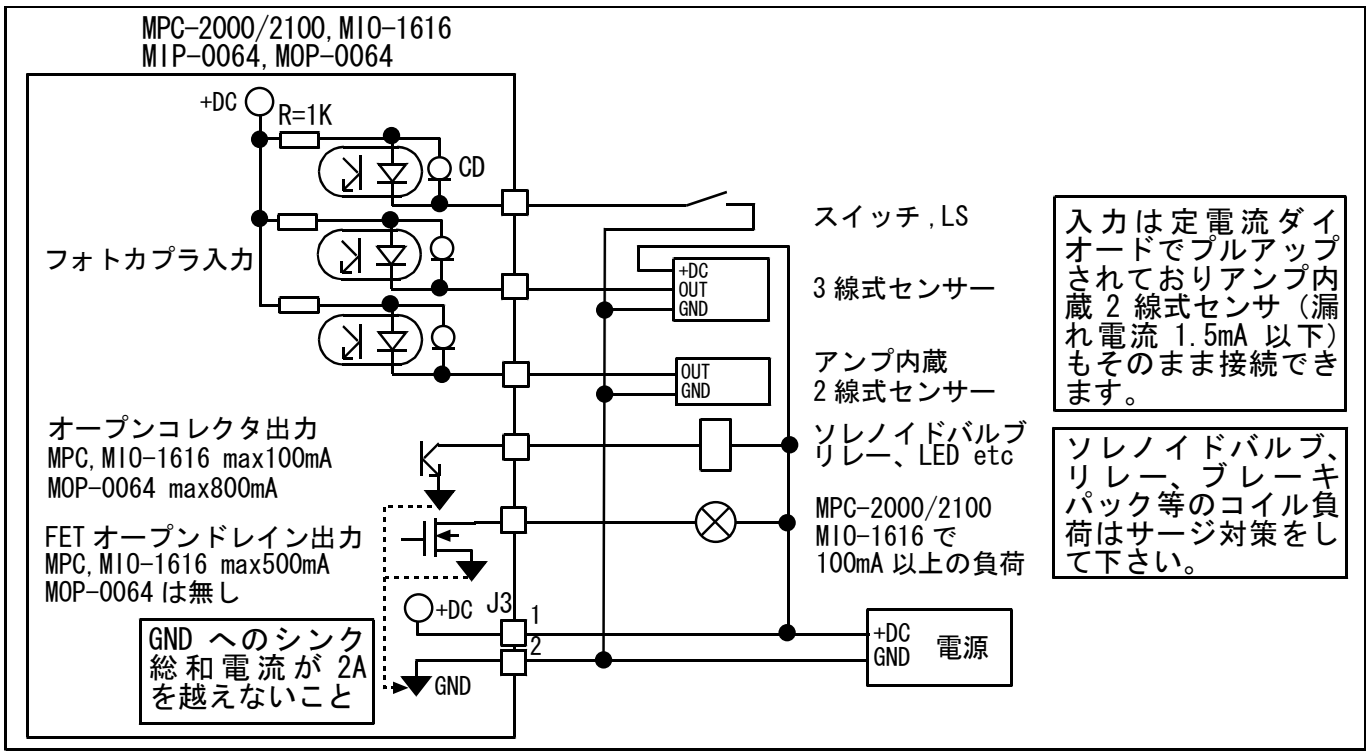
IN、OUT コネクタに I/O 機器を接続することができます。



IN		OUT	
ピン番号	ポート No	ピン番号	ポート No
1～8	200～207	1～8	4～11
9～16	NC	9～12	12～15

※この基板は MPC-XY03 専用の基板です。

◆ I/O 機器を接続したときの回路例



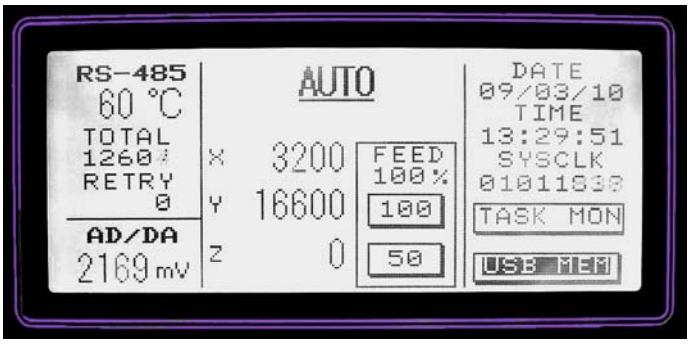
◆ QUATTRO2 とは

- 4 軸マイクロステップモータドライバです。各軸を独立で駆動します。
- 2 相ステッピングモータ ユニポーラ駆動、フォトカプラアイソレート入力。
- 専用ソフト(QCP.EXE)で各種設定を行います。
- 制御方法は 2pls(CW/CCW)または方向指示(PULSE/DIR) (4 軸一括設定)。
- 軸毎に分割数(1/2,1/4,1/8,1/16)、ドライブ電流、スリープ電流等の設定ができます。
- MPG-2314、MPG-2541 のパルスポートと直結できます。
- 動作電圧 DC12～24V 単一電源。(電圧が低い場合、高インピーダンスのモータは使えません)

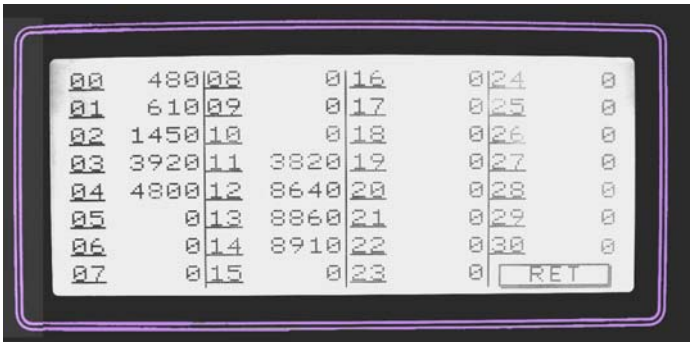
◆ AUTO モード

- セレクタを AUTO 側にすると自動運連を行います。
- RESET ボタンが点灯します(リセット状態)。

- ワークは左側 2 列(PALLET1)に置いてください。
- START を押すとパレット間のワーク搬送を開始します。
- STOP を押すと一時停止します。



- 各軸の現在座標値、調節計温度、AD/DA 電圧、MPC の時計、SYSCLK(経過時間)を表示します。SPEED は最高速の 100%または 50%です。
- TASK MON を押すとタスクモニタ画面に遷移します。
- リセット状態で USB MEM を押すと USB メモリ画面に遷移します。



- タスクモニタ画面です。各タスクの実行中の文番号が表示されます。

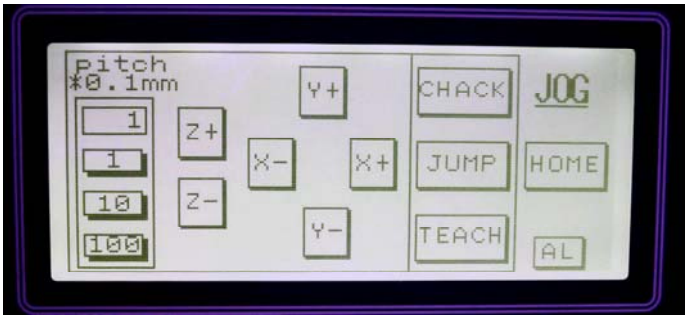
- USB メモリ画面です。USB メモリに保存されている点データファイルを読み込みます。

- ▲,▼　ファイル選択
- PLOAD & ACTION　点データ読み込み & 動作
- READ　ファイル名再読込
- TYPE　FTMW 画面にファイル内容を表示
- Ret　AUTO 画面に戻る

※タッチパネルの写真は色を変えています。

◆ MANU モード

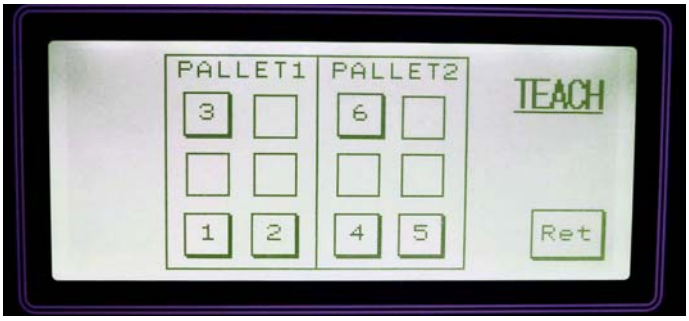
・ セレクタを MANU 側にするとティーチングが行えます。



・ JOG 移動を行います。

・ HOME	原点復帰
・ CHACK	ハンドメカチャック開閉
・ JUMP	JUMP 画面へ
・ TEACH	TEACH 画面へ
・ X+, X-, Y+, Y-, Z+, Z-	ジョグ移動
・ 1, 10, 100	移動量選択
・ AL	メカ調整画面へ

・ JUMP 画面で既にティーチングされている点へ移動できます。



・ TEACH 画面で現在点を教示できます。

※タッチパネルの写真は色を変えています。

＊ティーチング手順

・HOME で原点復帰

・JUMP で既にティーチングされている点へ移動

・各軸ジョグボタンで微動(ワークを置いてチャック開閉しながら位置を確認)

・位置が決まったら TEACH で点を指定

MPC-2000ファミリ

◆ MPC-2000 シリーズの製品概要です。

・メイン CPU
MPC-1000、MPC-2000 は小～中型(最大 8 スロットラック)、MPC-2100 は中～大型向きです。

MPC-1000	IN16、OUT16、RS-232 2CH (そのうち 1CH は RS-485 排他使用) ユーザープログラム 155KByte、点データ 7000 個、32 タスク パルス 2 軸(低速非同期)、USB メモリ、10bitAD×7CH、PWM LCD・バッテリバックアップ・カレンダ・時計 無し
MPC-2000	IN16、OUT16、RS-232 1CH ユーザープログラム 155KByte、点データ 7000 個、32 タスク LCD・バッテリバックアップ・カレンダ・時計 有り
MPC-2100	IN4、OUT4、RS-232 2CH ユーザープログラム 470KByte、点データ 16000 個、32 タスク、コプロ LCD・バッテリバックアップ・カレンダ・時計 有り

※上記 RS-232C の CH 数にプログラムポートは含まれません。

・拡張ボード

MIO-1616	入出力	IN16、OUT16、最大使用 8 枚
MIP-0064	入力	IN64 点、フォトカプラ入力、定電流ダイオードプルアップ 最大使用 6 枚
MOP-0064	出力	OUT64 点、RN1423 オープンコレクタ出力 最大使用 6 枚
MPG-2314	パルス発生	4 軸、直線補間(Max4Mpps)、円弧補間(Max2Mpps)、 エンコーダ・Up/Down カウンタ入力 2CH+オプション 2CH 最大使用 10 枚
MPG-2541	パルス発生	非同期 4 軸、Max400Kpps、最大使用 8 枚
MPC-MCOM	通信	RS-232C/RS485 拡張、USB メモリ、 最大使用 3 枚(MPC-1000 は 2 枚)
MPC-AD12	A/D,D/A	A/D8ch、D/A2ch (どちらも 12bit)、最大使用 2 枚
MPC-CUnet	ネットワーク	512byte メモリ共有、メール転送
USB-CUnet	ネットワーク	パソコン USB～CUnet インターフェース

・ケース、ラック

CASE-2S	ケース	2 スロット。縦置き用取り付け金具(オプション) : フランジ H
RACK-V4S	ラック	4 スロット縦置き
RACK-V8S	ラック	8 スロット縦置き
RACK-V16S	ラック	16 スロット縦置き

プログラム開発環境

ハードウェア

◆ パソコン　　Windows パソコン(USB 対応は 2000 以降)

◆ 接続(プログラム)ケーブル　　ケーブル DOS/V(DSUB9 ピン RS-232C ケーブル)
または USB-RS2(USB-SERIAL 変換)

ソフトウェア

FTMW　　ターミナルソフト

MPC と接続して編集・デバッグ、パソコンからの読み込み・保存を行います。
MPC の開発には必須のアプリです。
本文の対象は Ver-6.38z 以降です。
(ファイル名は「FTMW32.EXE」ですが、本文では「FTMW」としています)

＊＊＊

その他のソフトウェア

MPCED　　オフラインエディタ

MPC 専用のオフラインエディタです。制御文・ラベル・コメントを色分けします。

SYSLD2000　　システムローダ

MPC のバージョンアップ時に用います。フラッシュ ROM 内のシステムデータを書き換えます。

F2KCheck　　プログラム簡易チェッカー

判り難くなりがちな IF～END_IF、DO～LOOP の対応、ラベルの重複などをチェックします。

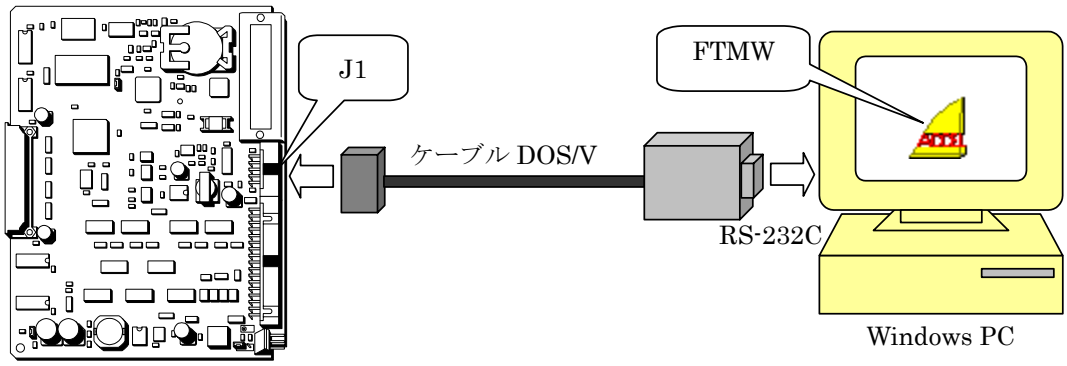
CUMON　　CUnet モニタ

CUnet グローバルメモリの確認・変更、CUnet メール送受信確認ツール。

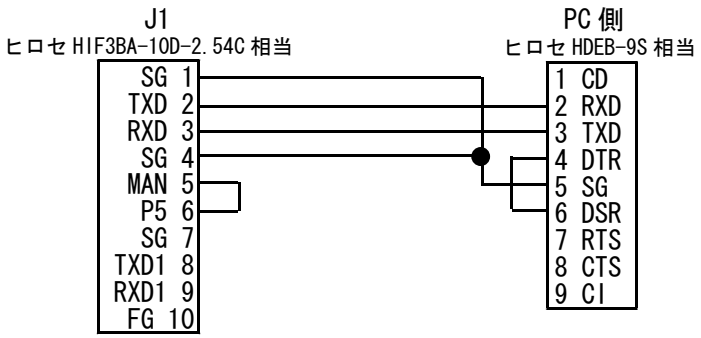
- MPC の開発環境は SetupDisk でインストールされます。
- SetupDisk は弊社ホームページから無償でダウンロードできます。
- 最初のインストールは必ず Setupdisk で行ってください。その後のアップデートは実行ファイル(*.EXE)の差し替えで Ok です。(最新版は web からダウンロードできます)。
- 標準セットアップフォルダは C:¥Program Files¥ACCEL です。

MPCとパソコンの接続

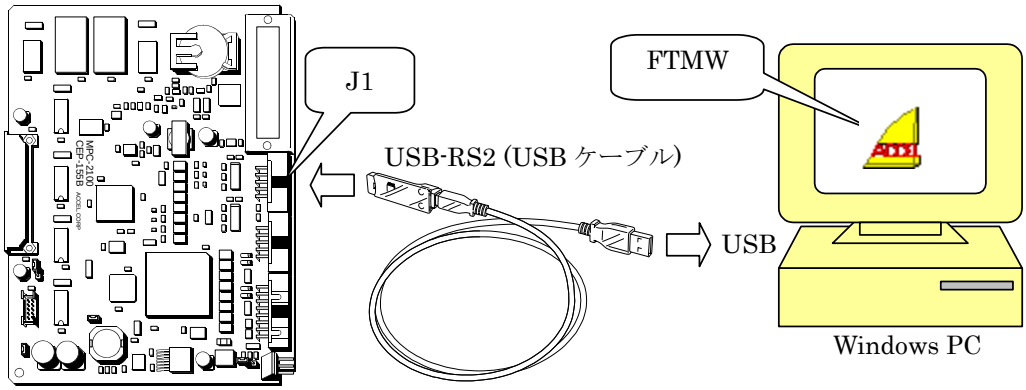
◆ パソコンの標準 COM ポートを使う場合(ケーブル DOS/V での接続)



(図は MPC-2000)



◆ パソコンの USB ポートを使う場合(USB-RS2 での接続)

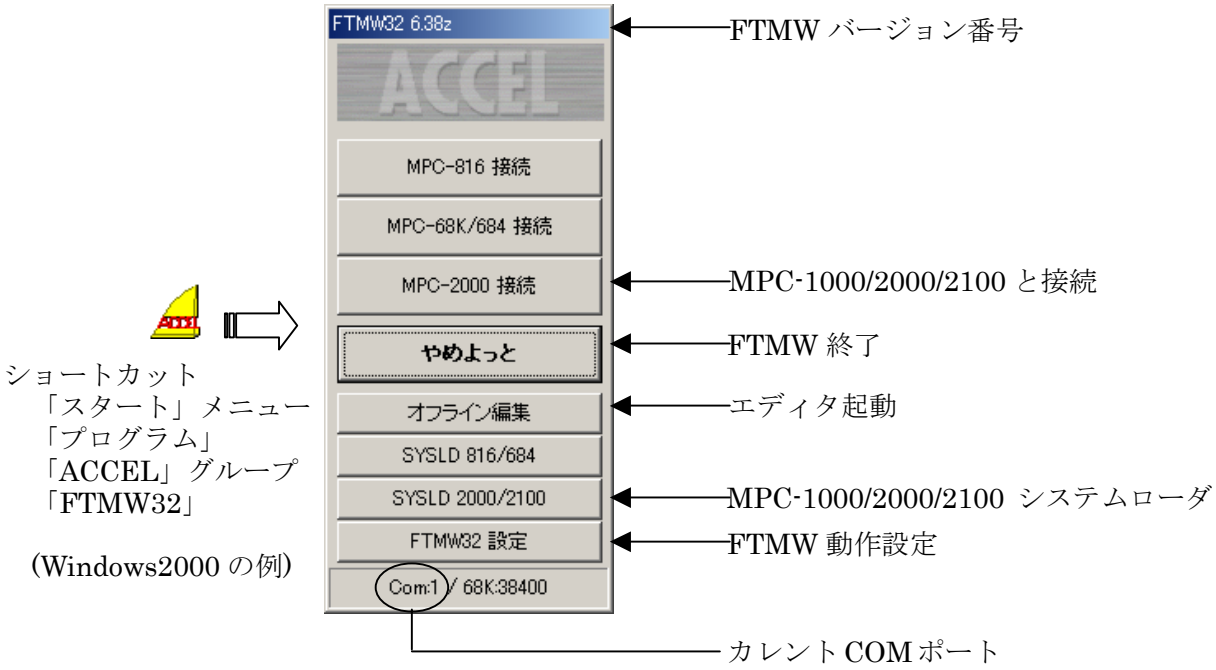


(図は MPC-2100)

USB-RS2 を使用するにはデバイスドライバのインストールが必要です

FTMW起動

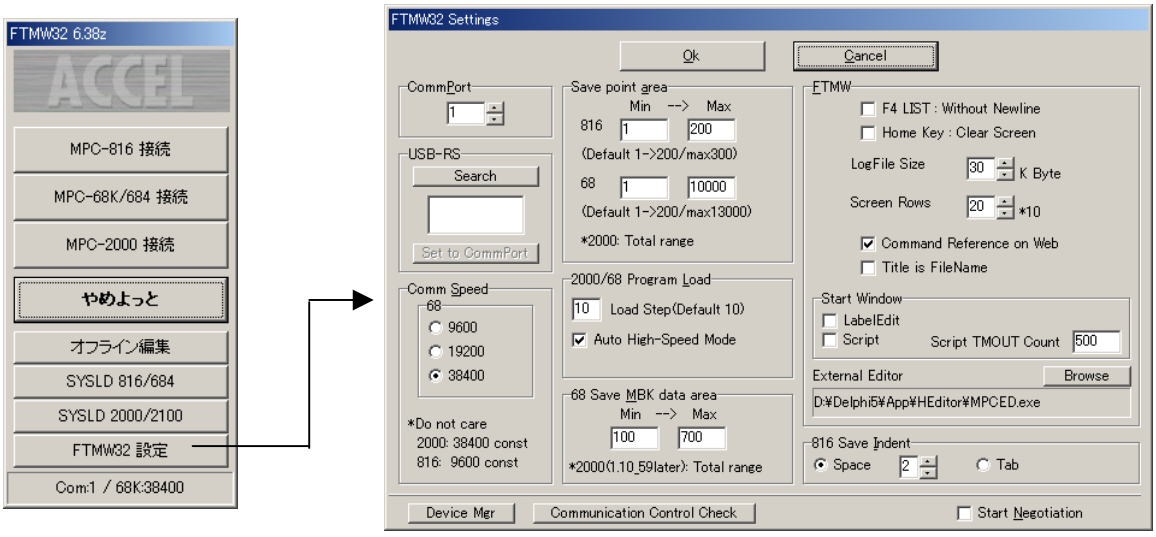
◆ FTMW のショートカットアイコンをクリックしてください。次のウィンドウが現れます。



◆ 「FTMW32 設定」画面

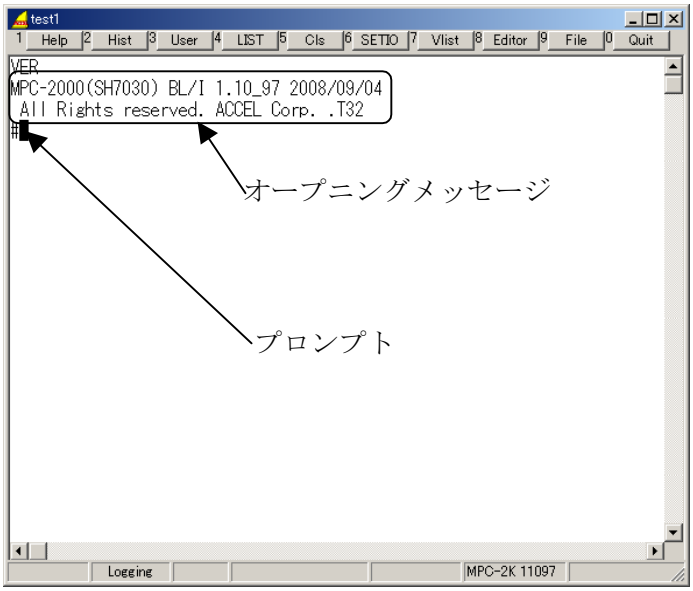
＊ Comm Port ：パソコンの標準 COM ポートの場合は 1~4 のいずれかになります。USB-RS2 を使う場合は「USB-RS」＞「Search」で検出できます。
ポート番号が判らないときは「Device Mgr」でデバイスマネージャを起動して「ポート(COM と LPT)」で確認して下さい。MPC-1000/2000/2100 の Comm Speed は 38400bps 固定です。

＊ Communication Control Check ： ウインドウズにインストールされている通信コントロールが日本語版か英語版かを調べます。日本語ウインドウズには日本語版の通信コントロールがインストールされていなければなりません。間違えて英語版をセットアップしてしまい、再度、日本語版をセットアップした場合などは確認してください。(チェックには ccc.exe が必要です。)



◆ MPC の電源を入れ「MPC-2000 接続」ボタンを押します。

◆ 編集画面にオープニングメッセージとプロンプトが表示されれば正常接続。



◆ オープニングメッセージの意味

このメッセージは FTMW 接続時や VER コマンドで表示されます。

・MPC-2100 の場合

```
MPC-2100(SH7030) BL/I 1.12_01 2009/11/09    /* MPC ハード(CPU),インタプリタバージョン,改版デート
 All Rights reserved. ACCEL Corp. .T32      /* コピーライト,タスク数
  [* and / performed before + or -]         /* 乗除演算優先モード
```

・MPC-1000 の場合

```
MPC-1000(SH7030) BL/I 1.12_03 2009/11/13
 All Rights reserved. ACCEL Corp. .T32
  [* and / performed before + or -]
```

・MPC-1000 で USB ポートが ON になっている場合

```
#ON_USB                                     /* USB メモリポートをオンする
#VER
MPC-1000(SH7030) BL/I 1.12_03 2009/11/13
 All Rights reserved. ACCEL Corp. .T32
  [* and / performed before + or -]
   +The USB Activated on TASK_29+           /* USB メモリポートがタスク 29 でアクティブ
```

・1.11_81 以前は整数演算中の*,/の優先演算がありませんでしたが、1.11_82 で暫定的に乗除演算優先対応しました。OLD コマンドで優先順位非対応モードになり、VER 表示からもなくなります。

```
#OLD
#VER
MPC-2100(SH7030) BL/I 1.12_01 2009/11/09
 All Rights reserved. ACCEL Corp. .T32
```

コマンド入力

- プロンプトの後にコマンドを入力して Enter すると、その場で実行します。これをダイレクトコマンド実行といいます。
- 殆どのコマンドはダイレクトに実行することもプログラムに記述することもできますが、メンテナンス・編集関係などでダイレクトコマンドでしか使えないものや制御文などでプログラムにしか書けないものもあります。

両方で使える	ダイレクトのみ	プログラムのみ
ON 0	LIST	GOTO
OFF 0	MPCINIT	GOSUB
PRINT A	ERASE	IF ~
MOV L	RUN	FOR ~ NEXT
など	など	など

```
#ON 0<Enter>          /* ダイレクト実行　10 ON 0<Enter> とすればプログラム
#GOTO 100<Enter>      /* ダイレクト実行しても何もおこらない
#10 MPCINIT<Enter>    /* このコマンドをプログラムするとプログラムが消える！
```

※本文中の<Enter>は、パソコンのキーボードの Enter キー押下を表します。

- XY03 での実行例

```
#ON 0                 /* フロントパネル緑 LED 点灯
#ON 1                 /* 黄 LED 点灯
#PRINT SW(192)        /* 緑 SW の状態確認
0                     /* 0=オフ
#PR SW(192)           /* 緑 SW を押しながら実行(PR は PRINT の短縮形)
1                     /* 1=オン
#PR IN(24)            /* DSW を'3'にしてパラレル入力
48
#PRX IN(24)           /* PRX は HEX 表示 10(Dec)=30(Hex)
00000030
#SETIO                /* 出力一括 OFF
#
```

初期化

◆ MPC の初期設定や実行時パラメータはフラッシュ ROM と S-RAM に記憶されます。
開発中の試行錯誤や搬送中(特に基板単体で)の静電気印加などでパラメータが狂うと動作不良になります。
次の時は初期化を励行して下さい。

＊ 基板単体で搬送したとき。
搬送中にダメージを受けることがあります。
メンテナンス等で基板単体にプログラムを入れて搬送する場合は、静電気、バッテリのショート・脱落、部品破損、結露などご注意ください。必ず帯電防止袋を使用して下さい。
＊ デバッグ中、挙動不審になったとき。
あれこれやっているうちにおかしくなったとき。
プログラムは良いはずなのにうまく動かない。(バグの可能性も追求してください)
etc
＊ システムのアップデート後。

◆ 初期化コマンド

＊ MPCINIT、ERASE
ダイレクトコマンドでこの 2 つを実行してください。
＊ JPN
MPCINIT でメッセージが英語モードになります。JPN で日本語モードになります。

```
#MPCINIT
#ERASE
*
#JPN
#
```

◆ 初期化時の注意

＊ 初期化するとプログラム、点データ、変数はクリアされます。
必要に応じてパソコンに保存、記録を取って下さい。

I/Oチェック

コマンドによるチェック

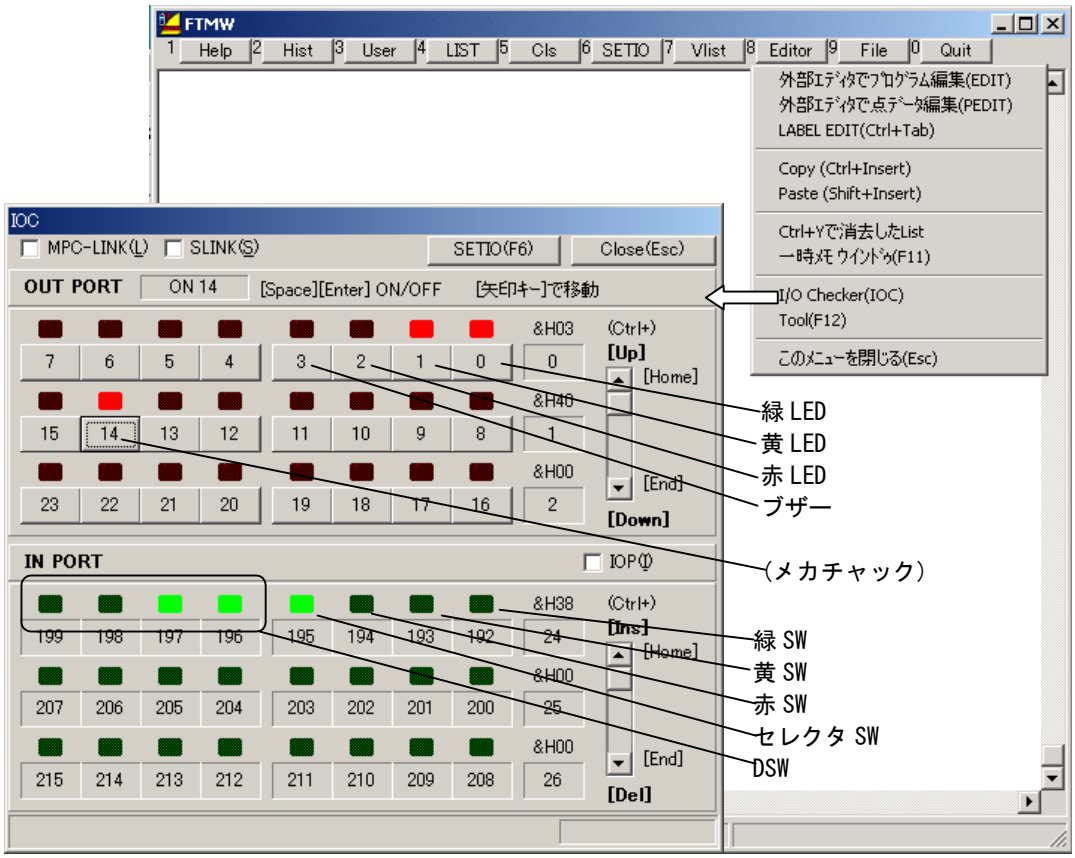
◆ ダイレクトコマンドによる I/O チェックの例です。

```
#ON 0                   /* フロントパネル緑 LED 点灯
#ON 1                   /* 黄 LED 点灯
#PRINT SW(192)          /* 緑 SW の状態確認
0                       /* 0=オフ
#PR SW(192)             /* 緑 SW を押しながら実行(PR は PRINT の省略形)
1                       /* 1=オン
#PR SW(195)             /* セレクタ SW 左側
0
#PR SW(195)             /* セレクタ SW 右側
1
#PR IN(24)              /* DSW を'3'にしてパラレル入力
56
#PRX IN(24)             /* PRX は HEX 表示 56(Dec)=38(Hex)
00000038
#SETIO                  /* 出力一括オフ
#
```

I/Oチェッカによるチェック

◆ まとめて見るなら [F8] I/O Checker で I/O チェッカを起動。

◆ IOC<Enter> としても I/O チェッカが起動します。



◆ 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
```

プログラム

プログラムとして入力するには

◆ 文番号を付けて記述するとプログラムになります。Enter キーを押して確定(MPC へ送信)です。エラーメッセージが出たら文法に誤りが無いか確認して再入力して下さい。

```
#NEW                              /* プログラム消去
#10 'サンプルプログラム           /* コメント文
20 DO                             /* 制御文
30 FOR I=0 TO 48                  /* 繰り返し
40 ON I : TIME 100                /* マルチステートメント
50 NEXT I
60 LOOP
70 pri I
        未定義コマンド            /* エラー行は入力されません
70 PR I                           /* 再入力
LIST 0                            /* 表示
10      'サンプルプログラム
20        DO
30         FOR   I=0 TO 48
40          ON   I : TIME   100
50         NEXT   I
60        LOOP
70        PRINT   I
#
```

マルチステートメント

◆ :(コロン)で区切って、1 行に複数のコマンドを書くことができます。
MPC 内で自動整形されるので長くなり過ぎないように注意してください。

```
#1000 WAIT SW(0)==0 : ON 0 : TIME 100 : OFF 0 : TIME 500
LIST 1000
1000      WAIT   SW(0)==0 : ON   0 : TIME   100 : OFF   0 : TIME   500  /* 入力時より長くなる
```

コメント

◆ '(シングルコーテーション)の後ろにコメントが書けます。

◆ プログラムの後ろにも書くこともできます。その場合は自動的にマルチステートメント化されます。

◆ コメントには日本語(全角、半角文字)も使えます。

```
#40 ON 0 'COMMENT
LIST 40
40        ON   0 :                          'COMMENT       /* マルチステートメントになる
```

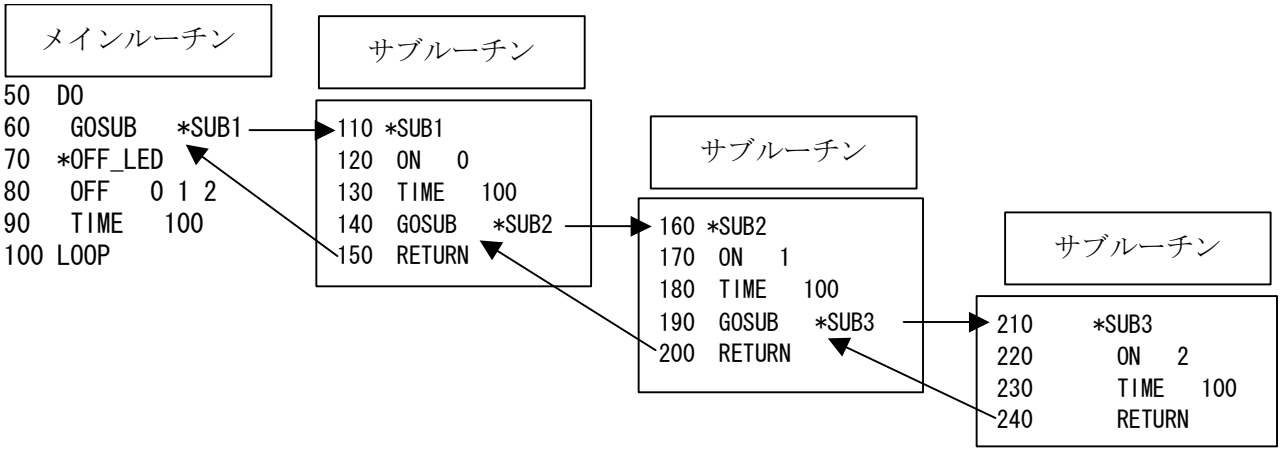
ラベル

◆ 先頭に＊が付いたものはラベルです。スペースは使えません。

```
10      *MAIN                               /* 飛び先のラベル名は重複禁止
20        IF   SW(0)==0 THEN  : GOTO  *MAIN : END_IF
```

サブルーチン

- 主なコマンド
  GOSUB、RETURN　　サブルーチンへ飛ぶ、サブルーチンから戻る
  _VAR、_RET_VAL　　サブルーチンへの引き数、サブルーチンからの戻り値
- 仕事単位でサブルーチン化して、それをメインルーチンから呼ぶようにするとプログラムが読み易くなります。
- GOSUB ばかりで RETURN が無いと "スタックが溢れました" エラーになります。



- GOSUB にはサブルーチンへの引き数を与えることができます。サブルーチン内では_VAR コマンドで値を取得します。
- RETURN の後ろにサブルーチンからの戻り値を与えられます。戻り値の取得は_RET_VAL です。
- ローカル変数と組み合わせるとタスク間でのサブルーチンの共有が可能となります。

```
10        OFF   0 1 2
20        TIME   500
30        GOSUB   *ON_LED 0 1 2 /* 引き数付きでサブルーチンコール
40        END
50      *ON_LED
60        _VAR   A_ B_ C_       /* 引き数を受け取る( _付きはローカル変数)
70        ON   A_
80        TIME   500
90        ON   B_
100       TIME   500
110       ON   C_
120       RETURN

10        GOSUB   *READ_DSW
20        _RET_VAL   D          /* 戻り値を受け取る
30        PRINT   D
40        END
50      *READ_DSW
60        DSW_=IN(24)/16        /* フロントパネル DSW 読み込み
70        RETURN   DSW_         /* DSW_は戻り値
```

繰り返し

◆ 主なコマンド

DO～LOOP、FOR～NEXT	繰り返し、順次処理
BREAK	繰り返しから抜ける

◆ 無限ループ

```
DO
  ON 0                    /* 緑 LED 点灯
  TIME 500
  OFF  0                  /* 緑 LED 消灯
  TIME 500
LOOP                      /* DO に戻る
```

◆ 10 回繰り返し

```
FOR CNT=1 TO 10
  ON 0
  TIME 500
  OFF  0
  TIME 500
NEXT CNT                  /* CNT を 1 つ加算して FOR に戻る
```

◆ 繰り返しから BREAK で抜ける

```
CNT=0
DO
  CNT=CNT+1
  IF CNT>10 THEN : BREAK : END_IF  /* DO~LOOP から抜ける
  ON 0
  TIME 500
  OFF  0
  TIME 500
LOOP
```

◆ 繰り返しから GOTO で抜ける

```
CNT=0
DO
  CNT=CNT+1
  IF CNT>10 THEN : GOTO *PASS : END_IF /* DO~LOOP から抜ける
  ON 0
  TIME 500
  OFF  0
  TIME 500
LOOP
*PASS
```

※FOR～NEXT から BREAK や GOTO で抜けることも出来ます。

条件分岐

◆ 主なコマンド

IF～THEN～[ELSE]～END_IF　　分岐
SELECT_CASE～END_SELECT　　数値による分岐

◆ IF 文。シンプルな条件判断向き。

```
IF SW(195)==1 THEN          /* フロントパネルセレクタ SW がオンならば
  GOTO *MANU
ELSE                        /* そうでなければ
  GOTO *AUTO
END_IF
*MANU
  OFF 0 : ON 1
  PRINT "MANUAL MODE"
  END
*AUTO
  ON 0 : OFF 1
  PRINT "AUTO MODE"
  END
```

◆ SELECT_CASE 文。いくつもの条件があるとき。

```
OFF 0 : OFF 1 : OFF 2       /* LED 消灯
DSW=IN(24)/16               /* フロントパネル DSW 読み込み
SELECT_CASE DSW             /* DSW の値を調べる
  CASE 0                    /* DSW=0 なら
    ON 0 : OFF 1 : OFF 2
  CASE 1                    /* DSW=1 なら
    OFF 0 : ON 1 : OFF 2
  CASE 2                    /* DSW=2 なら
    OFF 0 : OFF 1 : ON 2
  CASE_ELSE                 /* それ以外なら
    ON 3 : TIME 10 : OFF 3
END_SELECT
```

◆ SELECT_CASE の引数を VOID とすると、CASE 文独自の論理式を評価して実行します。複数の条件を調べなければならないとき、IF 文を羅列するより効率的です。

```
SELECT_CASE VOID
  CASE SW(192)==1           /* 緑 SW ON -> 緑 LED ON
    ON 0
    OFF 1 2
  CASE SW(193)==1           /* 黄 SW ON -> 黄 LED ON
    ON 1
    OFF 0 2
  CASE SW(194)==1           /* 赤 SW ON -> 赤 LED ON
    ON 2
    OFF 0 1
  CASE SW(195)==1           /* セレクタ SW 右側 -> LED OFF
    OFF 0 1 2
  CASE_ELSE
END_SELECT
```

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

◆ 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
```

◆ 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
```

◆ 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
```

◆ 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
```

◆ 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
```

◆ CODE→文字変換

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

*結果
ACCEL
```

◆ 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
```

◆ 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'
```

◆ 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
```

◆ 文字列複写(そのまんまコピー)

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

*結果
NOV15,2007
```

◆ 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
```

◆ 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
```

◆ 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
```

変数

◆ MPC の変数は自動変数です。パラメータにコマンド・関数・定数・予約文字列以外の文字列を与えると変数と解釈されます。C コンパイラのように変数名、型などの宣言は要りませんが、初期化はプログラムで行います。

```
10        GR_LED=0                        /* GR_LED は変数。0 に初期化
20         ON    GR_LED                   /* 変数を使って出力
```

◆ CONST で定数化します。

```
10         CONST    GR_LED 0             /* GR_LED は定数
20         ON    GR_LED                  /* 定数を使って出力
#GR_LED=1                                /* 定数の値を変えることはできません
        この変数は定数化されています
#
```

配列変数

◆ DIM 宣言で配列変数を確保します。

```
10         DIM    ARRY(100)    /* ARRY(0)～ARRY(99)の 100 個を確保
20         FOR    I=0 TO 99
30        ARRY(I)=I
40         NEXT    I
```

◆ DIM 配列は 2 次元も可能です。

```
10         DIM    ARRY(2,3)
20        ARRY(0,0)=1
30        ARRY(1,0)=2
```

◆ ポイントデータも一種の配列変数です。ポイントデータは FTMW でパソコンに保存・読み込みできます。現在の点データの内容は PLS コマンドで一覧表示します。

```
10         FOR    I=1 TO 2
20        X(I)=I
30        Y(I)=I
40         NEXT    I
50         END
#NEWP
#RUN
#PLS 0                                   /* 表示
P(1)     X= 1  Y= 1  U= 0  Z= 0
P(2)     X= 2  Y= 2  U= 0  Z= 0
P(3)     X= 0  Y= 0  U= 0  Z= 0
  (中略)
P(20)    X= 0  Y= 0  U= 0  Z= 0
#                                        /* "Q"ｷｰで表示終了。その他のキーで継続
```

ローカル変数

◆ ローカル変数とはタスク単位の変数です。同じ名前でもタスク毎に違うメモリエリアに割り当てられるので、１つのサブルーチンを複数のタスクで共有することができます。

```
A_=B_+C_                                /* '_'を付ければローカル変数
```

ON_ERROR処理

◆ ON_ERROR コマンドでインタプリタを止めずにエラーの処理と再開ができます。

・ 次は MPC-XY03 サンプルプログラムで行っているエラー処理です。
USB メモリから点データを読み込む時にエラーが発生すると *USB_MEM_ERR へジャンプしてタッチパネルにエラー情報を表示、再スタートボタン待ちになります。
ON_ERROR 指定しなければエラーが発生したタスクは停止しますが、この場合は停止しません。

```
5340      ON_ERROR  *USB_MEM_ERR      /* エラー発生時処理
5350      USB_PLOAD   FN$             /* USB メモリから点データ読み込み
5360      IF   err_==0 THEN           /* err_は予約変数。no error なら動作する。
5370       GOSUB   *USB_P_LIST
5380       S_MBK   3 993              /* タッチパネル "読み込みました" 表示
5390       TIME   1000
5400       S_MBK   0 993              /* タッチパネル 表示クリア
5410       GOSUB   *USB_ACT           /* 動作
5420      END_IF
5430      ON_ERROR   VOID             /* VOID すると err_ も リセット

----------

5640    *USB_MEM_ERR                  /* USB_PLOAD でエラーが起きたときの処理
5650      S_MBK   9 8                 /* これをしないと 2 回目以降にページが替わらない
5660      TIME   500                  /*    &H40 ページの RET ボタンが「全画面に戻る」のせいか?
5670      S_MBK   &H00000040 8        /* エラー表示画面
5680      WAIT   MBK(10)==&H00000040 /* ページが替わるのを待つ
5690      PRINT   "ERROR TASK=" TASKn "STEP=" err_&&H00FFFFFF "CODE=" err_>>24 "MSG=" ERR$(err_)
5700     MSG$="STEP="+STR$(err_&&H00FFFFFF)+" CODE="+STR$(err_>>24)+"          "
5710      S_MBK   "USB MEM READ ERROR  " 4000 20
5720      S_MBK   MSG$ 4020 20        /* タッチパネルにエラーが発生した行番号とコードを表示
5730      WAIT   MBK(10)<>&H00000040 /* [RET]ボタン(再スタート)押下待ち
5740      RESUME   _NEXT              /* エラー発生した行の次から再開
5750      END
```

演算

◆ 演算子

```
+    加算                          A=3+2          /* A=5
-    減算                          A=3-2          /* A=1
*    乗算                          A=3*2          /* A=6
/    除算                          A=3/2          /* A=1
%    剰余                          A=3%2          /* A=1
&    論理積(AND)                   A=3&2          /* A=2
|    論理和(OR)                    A=3|2          /* A=3
^    排他的論理和(XOR)             A=3^2          /* A=1
=    代入                          A=3            /* A=3
>    比較演算子 大きい             IF A>B THEN ...
<    比較演算子 小さい             IF A<B THEN ...
!=   比較演算子 等しくない         IF A!=B THEN ...
<>   比較演算子 等しくない         IF A<>B THEN ...
==   比較演算子 等しい             IF A==B THEN ...
>=   比較演算子 以上               IF A>=B THEN ...
<=   比較演算子 以下               IF A<=B THEN ...
,    上位下位合成演算              A=(3,2)        /* A=&H00030002
;    上位 8bit 下位 24bit 合成演算 A=(3;2)        /* A=&H03000002
~    上位ワードに合成              A=3~2          /* A=&H02000003
>>   右シフト                      A=A>>1         /* 下位側に 1bit
<<   左シフト                      A=A<<1         /* 上位側に 1bit
```

◆ 全て 4 バイト長整数です。小数点以下切り捨てです。

◆ MPC-2000 の算術演算では、加減算に対して乗除算優先となり、他は左から順に演算されます。
演算は式の前方から順に演算します。 優先順位が必要な場合は()で閉じてください。
A=1+2*3
この場合は、結果は 7 となります。1+2 を先に行う場合は、(1+2)*3 とします。
論理演算も同様です。
IF IN(0)&1==IN(1)&1 THEN
この例では、 IN(0)&1 と IN(1)&1 の比較を期待していると考えられますが、その場合は、
IF (IN(0)&1)==(IN(1)&1) THEN
と記述します。
また、算術演算と論理演算の区別はありません。比較演算子は、True の時に 1、False の時に 0 という結果を出力する演算です。
IF 文では、0 でなければ True、0 になれば False と扱います。このため、算術演算も IF 文で評価できます。
たとえば IF A==1 AND SW(0)==1 THEN は IF (A==1)&(SW(0)==1) THEN となります。

◆ MPC-2000 は浮動小数点演算をサポートしています。

```
' Get Pie
FLOAT   FP(6)=ACOS(SQR(3)/2)*6
FLOAT   FP(6)=(FP(6)-3)*10
PRINT   "PIE=3." FP(10000,6)
```

マルチタスク

- 主なコマンド
  FORK、PUASE、CONT、QUIT　　タスク起動、一時停止、継続、終了
- 複数の仕事を並列で実行できます
- タスク数は 32 本です。(0～31、ただし MEWNET で 31 を占有します)
- 子タスク同士の優先順位はありません。子タスク同士で起動、終了が可能です
- 実行中のタスクを FORK すると、そのタスクは先頭から再起動されます

```
DO
  WAIT SW(192)==1               /* 緑 SW オン待ち
  PRINT "起動"
  FORK 1 *TASK1                 /* タスク 1 起動

*CONT
  WAIT SW(193)==1               /* 黄 SW オン待ち
  PRINT "一時停止"
  PAUSE 1                       /* タスク 1 一時停止

  WAIT SW(192)==1 OR SW(194)==1 /* 緑 SW または赤 SW オン待ち
  IF SW(192)==1 THEN            /* 緑 SW なら
    PRINT "継続"
    CONT 1                      /* タスク 1 継続
    GOTO *CONT
  ELSE                          /* 赤 SW なら
    PRINT "終了"
    QUIT 1                      /* タスク 1 終了
    OFF 0
  END_IF
LOOP

*TASK1                          /* 緑 LED 点滅
  DO
    ON 0 : TIME 100
    OFF 0 : TIME 100
  LOOP
#RUN

起動            /* 緑 SW オン
一時停止        /* 黄 SW オン
継続            /* 緑 SW オン
一時停止        /* 黄 SW オン
終了            /* 赤 SW オン
起動            /* 緑 SW オン
```

デバッグ

- 主なコマンド
  RUN、MON、PRINT、LOG　　実行、停止行確認、文字表示、表示記録
  BREAK_POINT、CONT　　一時停止、再開
- キー操作
  Ctrl+A(または Ctrl+])、Ctrl+M、　　実行停止、停止行リスト

基本形(実行・停止・確認)

- MPC のデバッグの基本形は 実行・停止・確認 です。例えば、実行(RUN)して装置が止まったら停止(Ctrl+A)してプログラムの停止位置から停止原因を調べます。停止行を調べるには Ctrl+M です。

```
10        ON   0
20        WAIT   SW(192)==1
30        OFF   0
#RUN
                                        /* 実行して装置が止まったところで Ctrl+A でプログラム停止。
       *0    [20]                       /* タスク番号と停止行番号
#MON                                    /* MON コマンドでも停止行番号を表示
       *0    [20]
#                                       /* Ctrl+M して各タスクの停止行を確認
*0      20        WAIT   SW(192)==1     /* ﾀｽｸ 0 の 20 行で SW(192)オン待ち...ということが判る
```

※停止時に「!」マークが付くのは時間を浪費しているタスクです。全体の動作速度に影響することがあります。SWAP を入れてみてください。

```
10        FORK   1 *TASK1
20        END
100     *TASK1
110       DO
120       LOOP
#RUN

#        *0_   [20]      *1!   [110]    /* DO～LOOP で時間浪費
        !は時間浪費タスクです。

 ↓DO～LOOP 間に SWAP 追加
110       DO
115        SWAP                         /* SWAP を入れる
120       LOOP
#RUN
                                        /* Ctrl+A で停止
#        *0_   [20]      *1    [115]    /* !マークが無くなる
```

PRINTを仕込む

- プログラムの必要個所に PRINT 文を仕込んで変数や I/O の状態を表示させます。

```
10       C=0
20        DO
30       C=C+1
40         PRINT   "count=" C                    /* C の値をモニタする
50         IF   C>2 THEN  : BREAK   : END_IF
60        LOOP
70        PRINT   "おわり"
#RUN

count= 1
count= 2
count= 3
おわり
```

サブルーチン単位で実行

◆ 仕事単位でサブルーチン化しておくと部分的なデバッグができます。

```
10        GOSUB   *Z_HOME
20        END
30      *Z_HOME                          /* Z 軸が原点復帰するサブルーチン
40        PG   0
50        ACCEL   Z_A 10000 100 100
60        RMVS   Z_A -5000
70        WAIT   RR(Z_A)==0
80        SHOM   Z_A INO_ON
90        TMOUT   10000
100       HOME   0 0 0 50000
110       WAIT   RR(Z_A)==0
120       PRINT   "Z HOME"
130       RETURN
#RUN *Z_HOME                           /* ラベルを指定 RUN
30-
Z HOME

[130]   RETURN しすぎです               /* この場合エラー終了します。
#
```

BREAK_POINTコマンドで一時停止

◆ BREAK_POINT コマンドにより、8 個までの指定した文番号でプログラムを停止させることができます。(ラベル指定も可能です)。以下のようにプログラム番号を指定すると、指定行を表示します。その後指定行の文番号は、FTMW 上では反転表示されます。ブレークポイントは、順々に文番号を指定します。指定した文番号を解除する場合は、同じ番号を入力します。どの文番号が登録されているかは、BKP コマンドを引数なしで実行します。また、すべてのブレークポイントを解除するには、BKP 0 と入力します。

```
LIST
30        FORK   2 *bb
40        END
110     *bb
120       DO
130        FOR   i_=8 TO 15
140         ON   i_ : TIME   50 : OFF   i_
150        NEXT
160       LOOP
#BKP 110 140                                    /* ブレークポイントを設定する

110     *bb                                     /* 設定された文番号は反転表示になります
140       ON   i_ : TIME   50 : OFF   i_
#RUN

##
110     *bb     <02>                            /* ブレーク
?#                                              /* Enter 押下で次のブレークポイントまで実行
140         ON   i_ : TIME   50 : OFF   i_      <02>   /* ブレーク
?P i_                                           /* “P 変数(関数)” で内容確認
#PR  i_-> 8                                     /* ローカル変数 i_ は 8
E                                               /* プログラム停止
#
```

自動実行中の停止個所の確認方法

◆ 自動実行中に停止した場合、電源を切らずにそのままケーブルを差込み FTMW と接続してください。

```
<<装置が止まった！なぜ？-→プログラムケーブルを差込み FTMW と接続>>
VER
MPC-2000(SH7030) BL/I 1.10_97 2008/09/04
All Rights reserved. ACCEL Corp. .T3        /* MPC と接続完了
#MON                                        /* MON コマンドでも停止位置が判る
        *0    [20]
#                                           /* Ctrl+M で各タスクのリストを表示
TASK0 20          WAIT   SW(192)==1         /* タスク 0 が 20 行で SW(192)オン待ち。

LIST
10        ON   0
20        WAIT   SW(192)==1
30        OFF   0
#
```

プログラムポートの出力記録

◆ LOG<Enter>とすると RS-232C プログラムポートの出力記録を表示します。これにより自動実行時のランタイムエラーの事後確認ができます。

◆ 記録メモリは 1k バイトのリングバッファです。

◆ LOG 0<Enter>とするとバッファをクリアします。

```
10        FOR   I=1 TO 2
20         PRINT   I
30        NEXT   I
#LOG 0                          /* LOG データクリア
#RUN                            /* 実行

1                               /* プログラム実行中の PRINT 表示
2
#LOG                            /* LOG データを見る
1                               /* 記録されたデータ

2                               /* 記録されたデータ

#LOG                            /* これは最後に入力した"LOG"のエコーバック(無視してください)

#
```

特殊なプログラム

- タスク 0 を END で終了すると FTMW にプロンプトが返ってきます。この状態で MPC へコマンドを発行することが可能になり、リアルタイムで変数などのモニタ、実行位置の確認ができるようになります。
- 注意

  ＊プログラム中に PRINT 表示があると表示が重なります。

  ＊プログラムを変更すると停止します。

```
10        FORK   1 *GR_LED_FLICK
20        END                              /* タスク 0 を END で終わらしてしまう
30      *GR_LED_FLICK
40        DO
50         ON   0
60         TIME   100
70         OFF   0
80         TIME   100
90        LOOP
#RUN                                       /* 実行

#MON                                       /* プロンプトが返ってくる。コマンド入力ができる。
        *0    [-1]      *1    [80]        /* 子タスクは動いている
#MON
        *0    [-1]      *1    [60]        /* 子タスクは動いている
#PRINT SW(192)                             /* 入力の状態も見ることができる
0
#PRINT SW(192)
1
#
```

FTMWでのプログラム編集

◆ FTMW で現在 MPC に入っているプログラムを編集する際、頻繁に行う操作を解説します。

LISTの表示

◆ 最も頻繁に使用するのが LIST コマンドです。

＊書式

LIST [arg1 arg2]

arg1: 開始文番号或は開始ラベル

arg2: 表示行数

◆ LIST だけでも実行できます(下記①)。その場合は前回の続が表示されます。

◆ 第 1 パラメータに表示開始位置を文番号またはラベルで指定できます(下記②③)。

◆ 第 2 パラメータに表示する行数を指定できます(下記④)。以後この行数は保持されます。

◆ LIST 0 とすると最初から表示します(下記⑤)。

①
```
#LIST   /* 開始位置、行数指定無し
10        GOSUB   *READ_DSW
20        _RET_VAL   D
30        PRINT   D
40        END
50      *READ_DSW
#
```

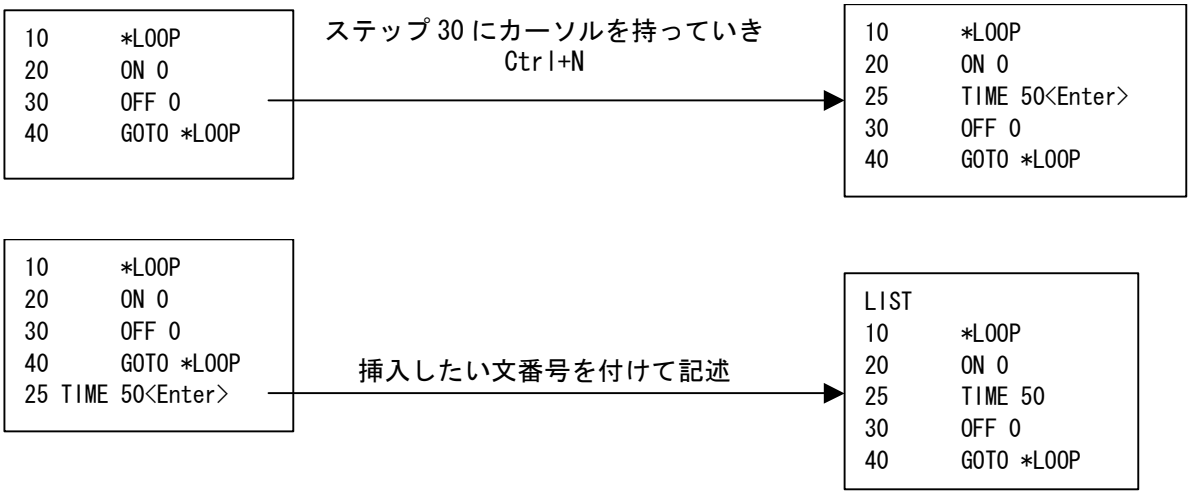
②
```
#LIST 40    /* 開始位置=文番号指定
40        END
50      *READ_DSW
60       DSW_=IN(24)/16
70        RETURN   DSW_
#
```

③
```
#LIST *READ_DSW   /* 開始位置=ラベル指定
50      *READ_DSW
60       DSW_=IN(24)/16
70        RETURN   DSW_
#
```

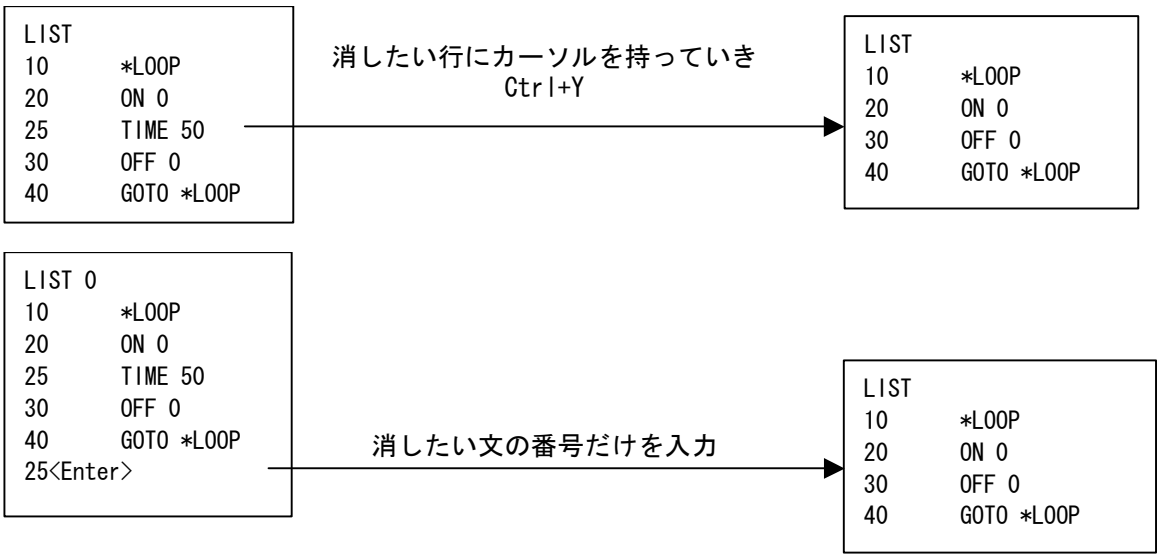
④
```
#LIST *READ_DSW 2   /* 開始位置=ラベル指定、表示行数指定
50      *READ_DSW
60       DSW_=IN(24)/16
#
```

⑤
```
#LIST 0 5   /*開始位置=先頭、表示行数指定
10        GOSUB   *READ_DSW
20        _RET_VAL   D
30        PRINT   D
40        END
50      *READ_DSW
#
```

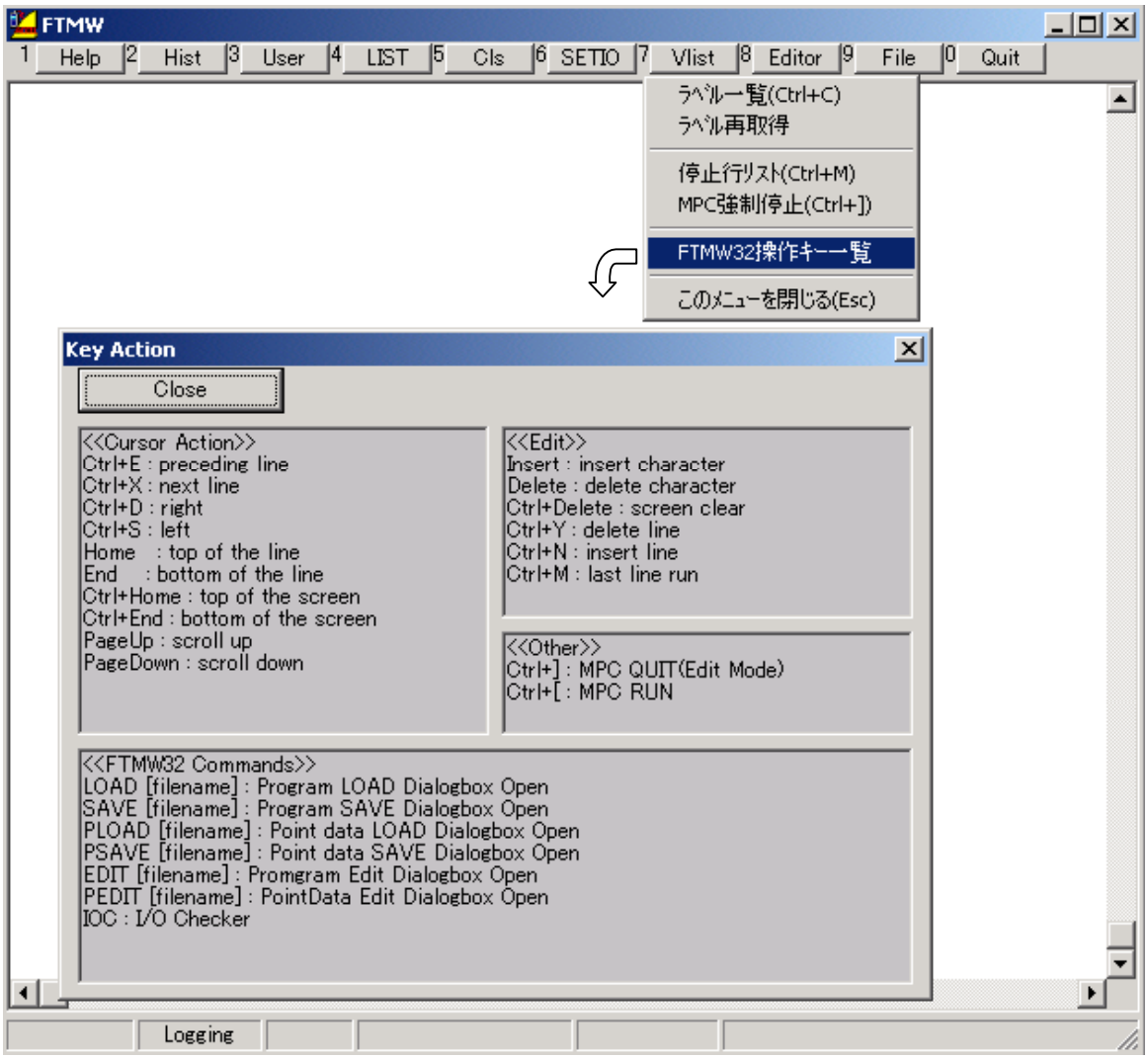
行の挿入


10 *LOOP
20 ON 0
30 OFF 0
40 GOTO *LOOP
ステップ 30 にカーソルを持っていき
Ctrl+N
10 *LOOP
20 ON 0
25 TIME 50<Enter>
30 OFF 0
40 GOTO *LOOP
10 *LOOP
20 ON 0
30 OFF 0
40 GOTO *LOOP
25 TIME 50<Enter>
挿入したい文番号を付けて記述
LIST
10 *LOOP
20 ON 0
25 TIME 50
30 OFF 0
40 GOTO *LOOP

行の削除


LIST
10 *LOOP
20 ON 0
25 TIME 50
30 OFF 0
40 GOTO *LOOP
消したい行にカーソルを持っていき
Ctrl+Y
LIST
10 *LOOP
20 ON 0
30 OFF 0
40 GOTO *LOOP
LIST 0
10 *LOOP
20 ON 0
25 TIME 50
30 OFF 0
40 GOTO *LOOP
25<Enter>
消したい文の番号だけを入力
LIST
10 *LOOP
20 ON 0
30 OFF 0
40 GOTO *LOOP

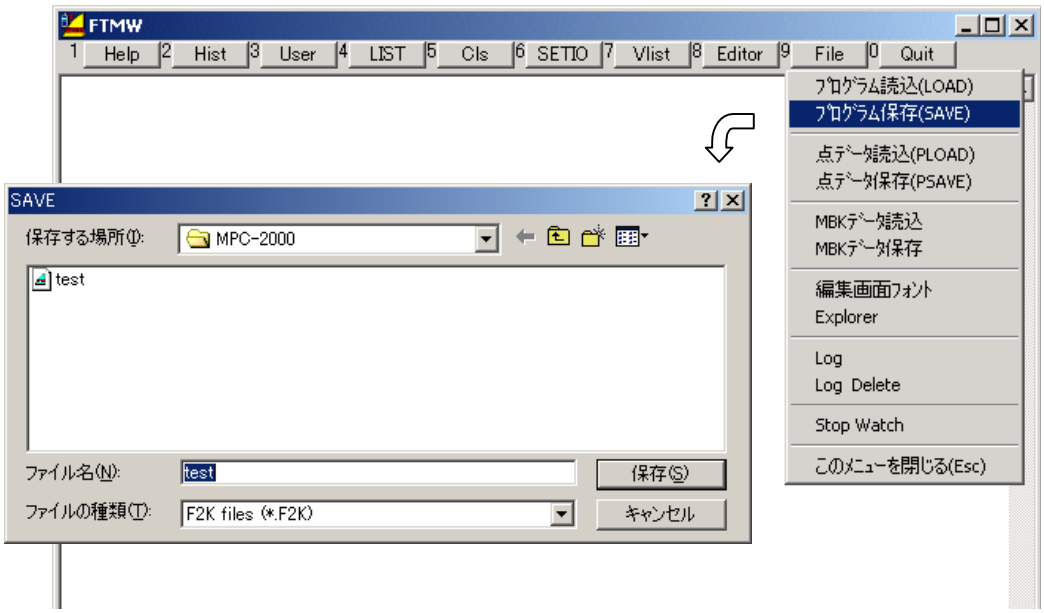
その他のキー操作


FTMW
1 Help
2 Hist
3 User
4 LIST
5 Cls
6 SETIO
7 Vlist
8 Editor
9 File
0 Quit
ﾗﾍﾞﾙ一覧(Ctrl+C)
ﾗﾍﾞﾙ再取得
停止行ﾘｽﾄ(Ctrl+M)
MPC強制停止(Ctrl+])
FTMW32操作ｷｰ一覧
このﾒﾆｭｰを閉じる(Esc)
Key Action
Close
<<Cursor Action>>
Ctrl+E : preceding line
Ctrl+X : next line
Ctrl+D : right
Ctrl+S : left
Home : top of the line
End : bottom of the line
Ctrl+Home : top of the screen
Ctrl+End : bottom of the screen
PageUp : scroll up
PageDown : scroll down
<<Edit>>
Insert : insert character
Delete : delete character
Ctrl+Delete : screen clear
Ctrl+Y : delete line
Ctrl+N : insert line
Ctrl+M : last line run
<<Other>>
Ctrl+] : MPC QUIT(Edit Mode)
Ctrl+[: MPC RUN
<<FTMW32 Commands>>
LOAD [filename] : Program LOAD Dialogbox Open
SAVE [filename] : Program SAVE Dialogbox Open
PLOAD [filename] : Point data LOAD Dialogbox Open
PSAVE [filename] : Point data SAVE Dialogbox Open
EDIT [filename] : Promgram Edit Dialogbox Open
PEDIT [filename] : PointData Edit Dialogbox Open
IOC : I/O Checker
Logging

プログラムの保存・読み込み

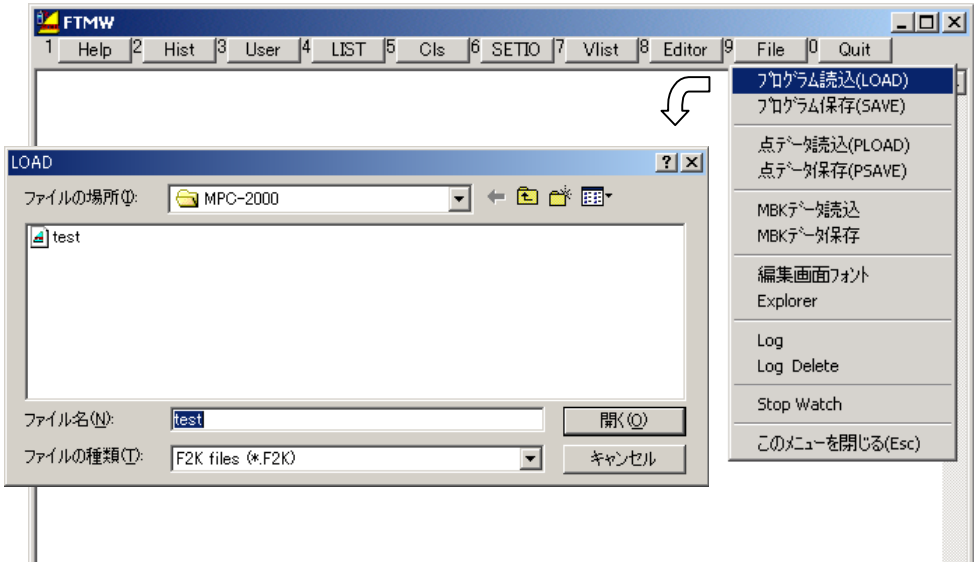
保存

- F9 「プログラム保存」でパソコンに保存します。
- 拡張子‘F2K’で保存されます。
- 保存したプログラムには文番号はありません。



読み込み

- F9 「プログラム読込」でパソコンからプログラムを読み込みます。
- 文番号は初期状態で 10 間隔です。60000 ステップを越えると自動的に 5 で RENUM します。



オフライン作成

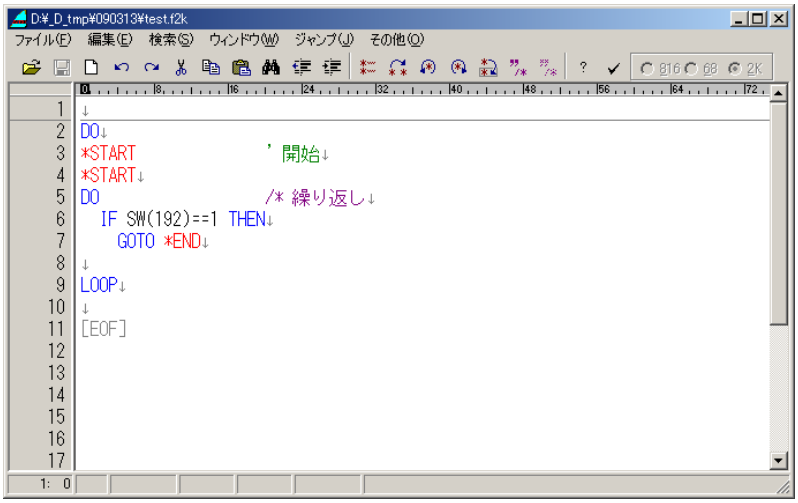
- FTMW は MPC 内のデータを直接操作するもので、接続していないと使うことはできませんが、プログラムはオフラインで作成することができます。
- エディタをご用意ください。MPCED は MPC 専用ですが、汎用のエディタやワープロでも使えます。その場合、プログラムはテキストデータ、拡張子を”F2K”として保存して下さい。
- プログラムのエラーは MPC に読み込み、実行するまではわかりません。

印刷

- FTMW には印刷機能はありません。パソコンに保存したファイルを MPCED などのエディタやワープロソフトで印刷してください。

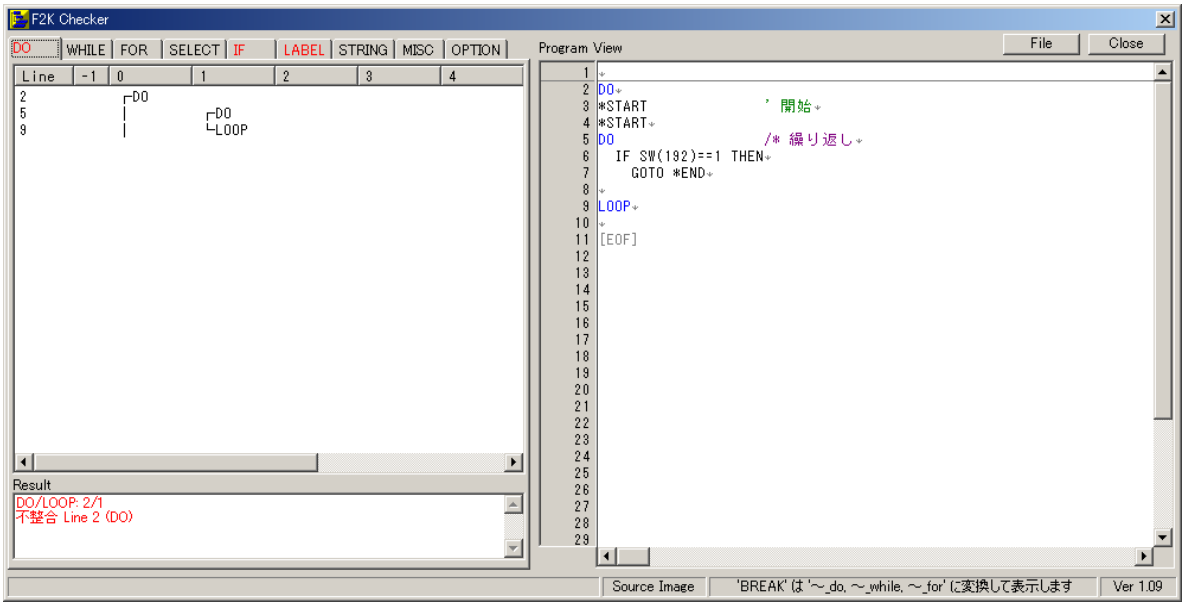
オフラインでプログラム作成・編集

- FTMW は MPC と接続して使用するもので、パソコンに保存したプログラムの編集や、作成はできません。オフラインでのプログラミングはエディタを使用します。エディタは、一般的にテキストエディタと呼ばれるもので、市販品、フリーウェアや Windows のメモ帳でも作成できます。
- MPC 専用のエディタとして「MPCED」があります。制御文などの色分け、ラベルジャンプ、一括コメント/アンコメントなどの機能があります。
- インターネットに接続できる環境なら、(HELP を参照する要領で)コマンド上にカーソルを置き F1 キー押下で MPC-2000 サイトのコマンドリファレンスを直接参照できます。



- プログラムが大きくなると DO～LOOP や IF～END_IF などの対応が判り難くなりがちです。そんなときは簡易チェッカー「F2KCheck」を使ってみてください。MPCED のチェックボタンで起動します。

  - 例えば上のプログラムは DO～LOOP、IF～END_IF が正しく対応していません。またラベルの重複があり、GOTO の行き先がありません。これをチェッカーでみると次のようになります。赤いタブが警告を表します。



※これらは単純な数合わせです。プログラムの正常動作を保証するものではありません。

I/O制御

- I/O はビット単位、またはバイト単位(バンク)で扱うことができます。
- ビットでは 0 か 1 の状態しかありませんが、バンクでは 0~255(&HFF)のデータを持ちます。主に、ビット操作はソレノイドバルブ・リレー・スイッチなどの機器の単体制御、バイト操作は DSW 読み込みやシーケンサなどの外部機器とのデータ授受に利用されます。
- バンクとはバイト単位の表現です。入出力 0~7 がバンク 0、8~15 がバンク 1、16～23 がバンク 2 というようになります

ビット処理

- 主なコマンド
  SW、ON、OFF　　ビット読み書き

```
ON 0                                        /* フロントパネル緑 LED 点灯
OFF 1                                       /* フロントパネル黄 LED 消灯
WAIT SW(192)==1                             /* フロントパネル緑 SW がオンになるまで待つ
IF SW(195)==1 THEN : GOTO *LABEL : END_IF /* もしもセレクタ SW がオンならば...
ON -1                                       /* メモリ I/O -1 をオンする
WAIT SW(-2)==0                              /* メモリ I/O -2 がオフになるのを待つ
```

バイト処理

- 主なコマンド
  IN、OUT　　バイト読み書き

```
OUT &H5 0                                   /* 赤 LED と緑 LED 点灯
A=IN(24)                                    /* フロントパネル入力の状態を変数 A に入れる
OUT A B                                     /* A の値をバンク B にパラレル出力
IF IN(24)==&H70 THEN : GOTO *LABEL : END_IF /* 条件分岐
OUT &HFF -1                                 /* メモリ I/O のバンク-1 を&HFF にする
WAIT IN(-1)==&HAA                           /* メモリ I/O のバンク-1 が&HAA になるのを待つ
```

※IN、OUT の基本形は 1 バイトですが~Wrd や~Lng を付加して 2 バイト長 4 バイト長の読み書きもできます。

メモリI/O

- 負の I/O 番号を与えるとメモリ I/O として扱います。
- タスク間のインターロックなどに使います。

```
ON -1                       /* ビット-1 をオン
WAIT SW(-1)==1              /* ビット-1 がオンになるまで待つ
OUT 255 -2                  /* バンク-2(ビットでは-9～-16)に 255 を出力
```

パルス発生

※これ以降のサンプルプログラムは MPG-2314 で作成したものです。

初期設定

- 主なコマンド

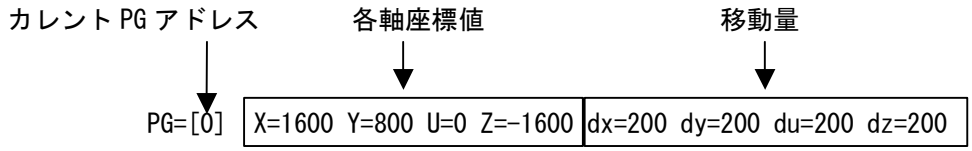
  PG、ACCEL、FEED、INSET　　PG 選択、速度設定、入力設定
- MPG-2314 を搭載しただけでは正常にパルスは出ません。初期設定が必要です。
- 最初に PG コマンドでタスクに MPG を引き当てます。次に ACCEL 等で初期設定をします。
- ダイレクトコマンドでも可能ですが、最終的にプログラムに反映させてください。
- 設定例

```
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、T(TEACH)、PLS　　PG 選択、ティーチモード、点データ一覧表示
- 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

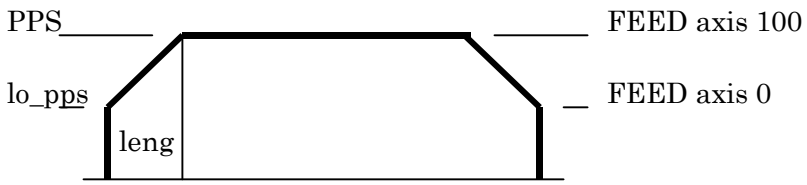
◆ 主なコマンド

ACCEL、FEED　　最高速度・加減速・最低速度設定、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

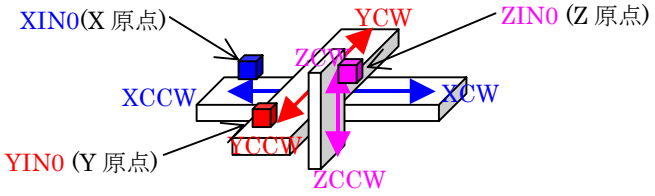


原点復帰

◆ 主なコマンド

SHOM、HOME、HPT　　原点入力設定、原点復帰動作、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。



◆ 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
```

◆ XY 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
```

◆ 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
```

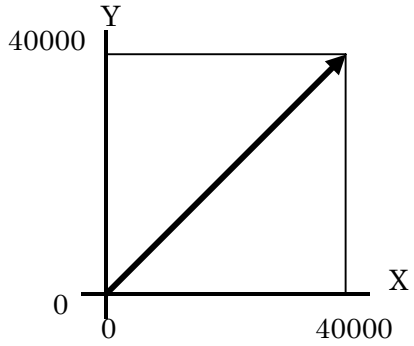
※これらのサブルーチンは後述のサンプルでも使用しています。

絶対座標移動

◆ 主なコマンド

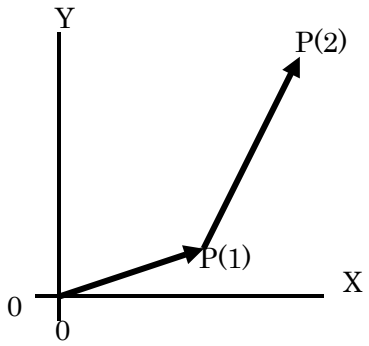
MOVL、MOVS　　直線補間移動、単軸移動

◆ 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。



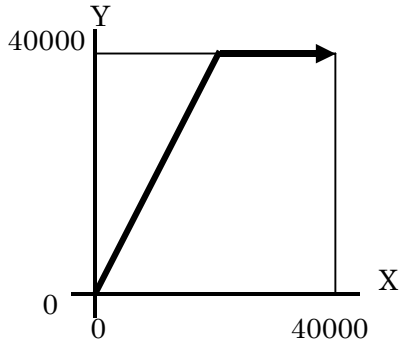

```
GOSUB *Z_HOME                  /* 前述の原点復帰サブルーチン
GOSUB *XY_HOME

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。



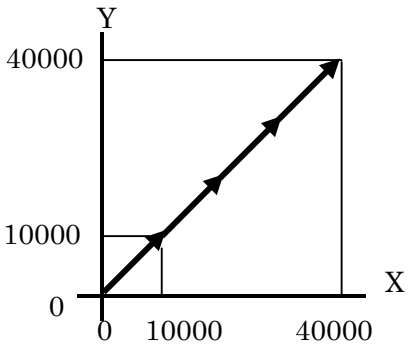

```
GOSUB *Z_HOME
GOSUB *XY_HOME

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

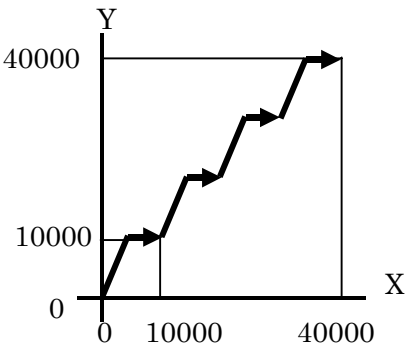
RMVL、RMVS　　直線補間移動、単軸移動

◆ 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。



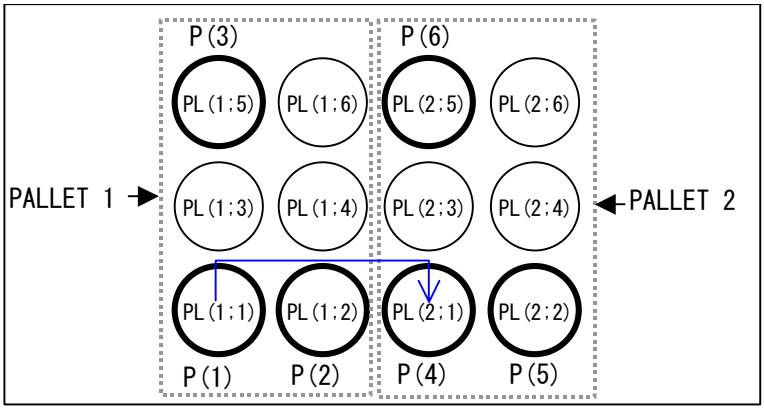

```
GOSUB *Z_HOME
GOSUB *XY_HOME

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、PL　　パレット宣言、作業点

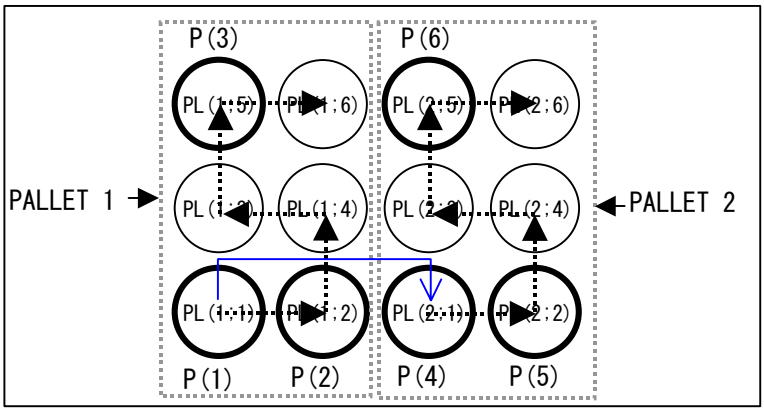
◆ パレット間の移動に利用します。角の 3 点と行列数からパレット上の作業点 PL を算出します。




```
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、INSET　　パルス停止、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。

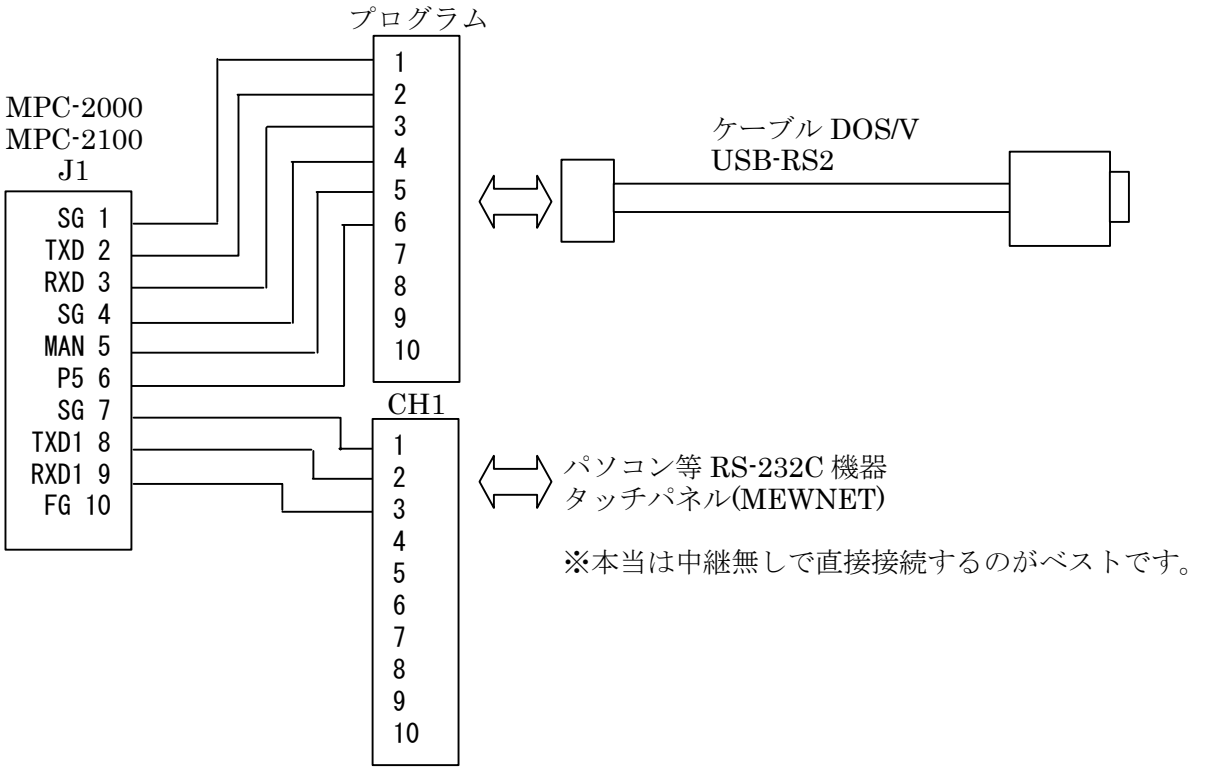
移動中に X 軸アラーム(J6 コネクタ 13 番ピン)または Y 軸アラーム(同 14 番ピン)がオンになると両軸は即停止します。

```
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
```

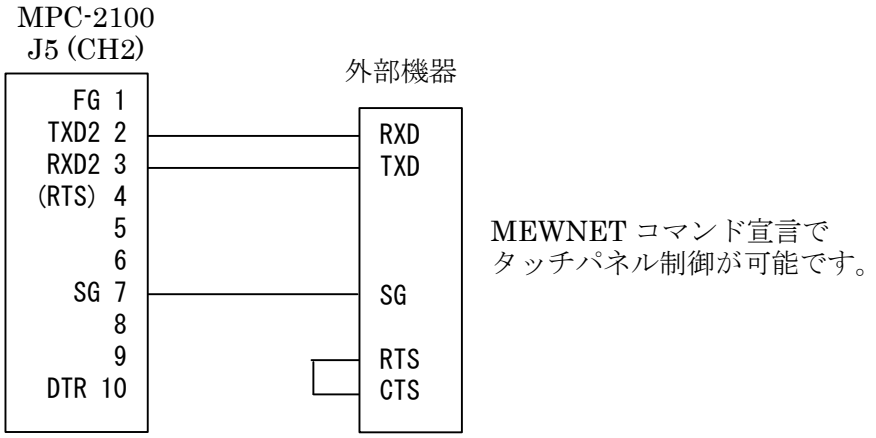
※ALM などの MPG-2314 入力のチェックは INCHK コマンド。

シリアル通信

- MPC にはプログラミング用の他に、ユーザープログラムで制御可能なシリアルポートがあります。RS-232C ユーザーポートはコマンド切り替えにより MEWNET プロトコル(準拠)をサポートし、タッチパネルとダイレクトアクセスが可能です。
- J1 コネクタには 2 つの RS-232C ポートが入っているので、通信を行う場合は下記のような分岐・中継ケーブルを作成してください。



- MPC-2100 は J5 コネクタにも RS-232C ユーザーポートがあります。3 線クロスで外部機器と接続します。



- RS-232C ポートの増設や RS-422/RS-485 接続の場合は MRS-MCOM を使用します。

RS-232C機器との接続

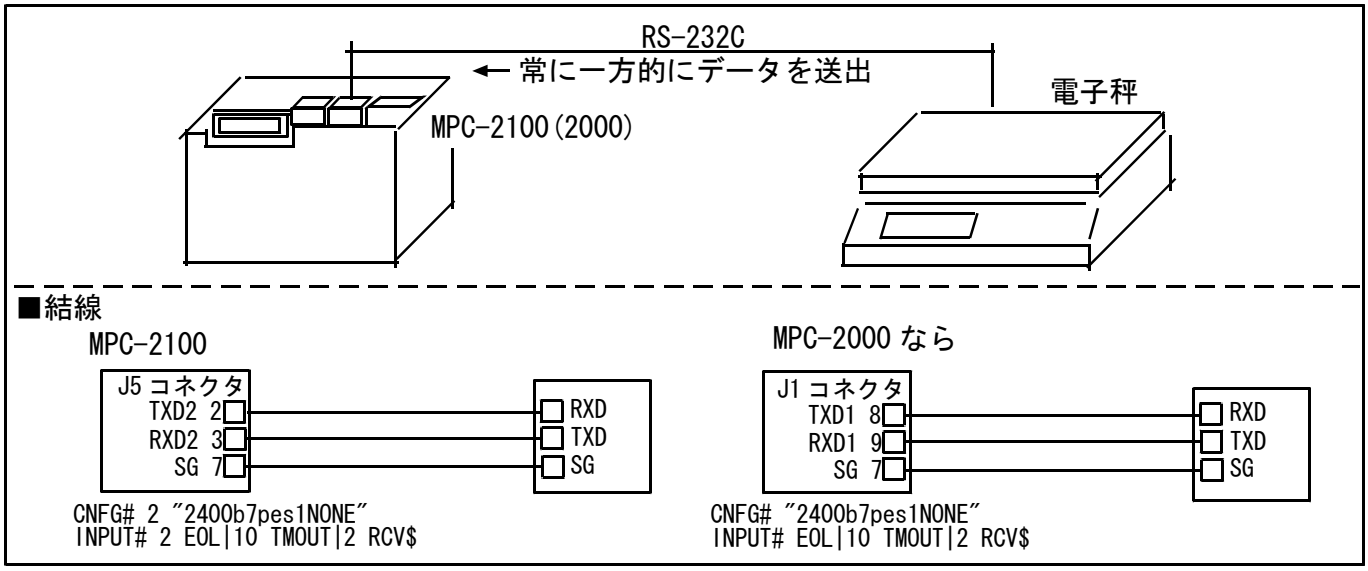
◆ 主なコマンド

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です。

```
/*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
```

◆ 電子秤との続例です。



・ 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$="〇"
      CASE "US" : RESULT$="△"
      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 〇 0 ,0 g
 2 0 〇 0 ,0 g
 3 0 〇 0 ,0 g
 4 0 〇 0 ,0 g
 5 0 〇 0 ,0 g
 6 0 〇 0 ,0 g
 7 0 〇 0 ,0 g
 8 0 〇 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 〇 117 ,3 g
18 0 〇 117 ,3 g
```

(see also: アプリケーションノート an2k-005)

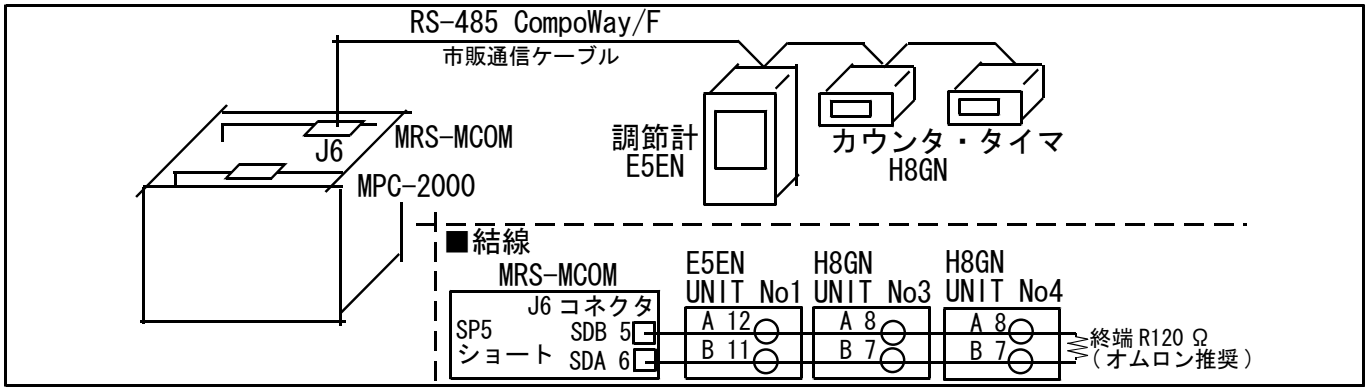
RS-485機器との接続

◆ 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(レスポンスフレーム 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。

- 送信手順
  1) COMPOWAY コマンドで送信するテキストを構築します。
  2) 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。
- 受信手順
  1) 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。
  2) 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。

- 下記は 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します

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)

タッチパネル接続

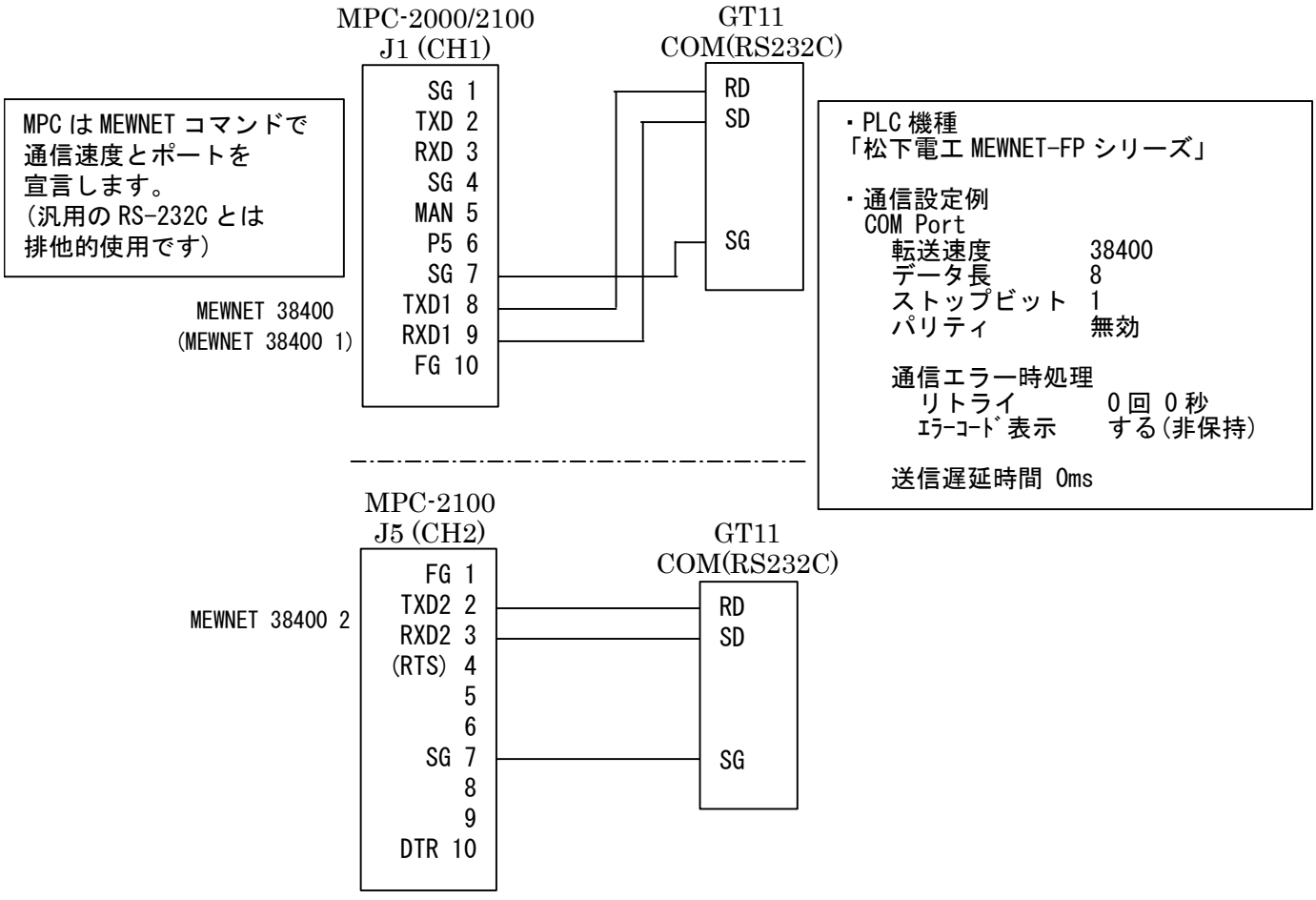
◆ 主なコマンド

MEWNET	機能切り替え
SW、IN、ON、OFF、OUT	I/O エリア (R0～R099F) 読み書き
MBK、S_MBK	データエリア(DT0～DT7899) 読み書き

※MPC のコマンド引き数は、I/O エリア：70000～79915、データエリア：0~7899 です。
※MEWNET はタスク 31 を占有するので他の制御には使用できなくなります。

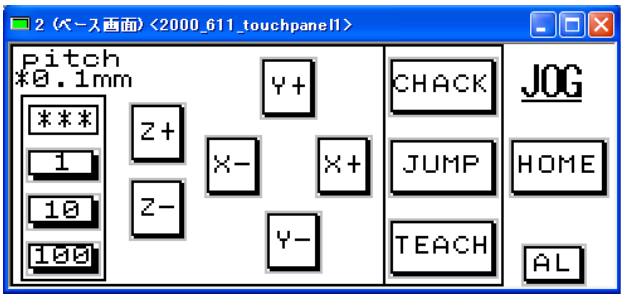
・例)
スイッチ部品 R0 を読むには SW(70000)
ランプ部品 R8 をオン/オフするには ON 70008 / OFF 70008
データ部品 DT101 を読むには MBK(101)
データ部品 DT102 に 123 を書き込むには S_MBK 123 102
ページを変えるには S_MBK pageno 8

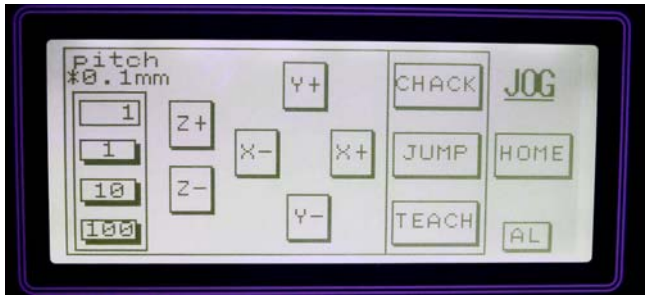
◆ 松下電工 GT11 との接続例



◆ 松下電工 GT11 で作成したティーチング画面の例です。

・ 画面番号：2(ベース画面)





※写真は色を変えています。

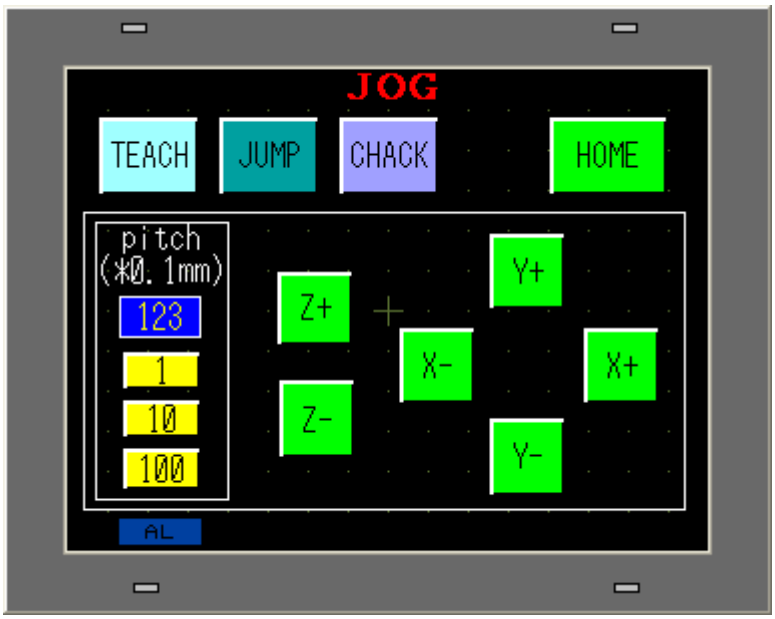
ページ切り替え S_MBK 2 8

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)	

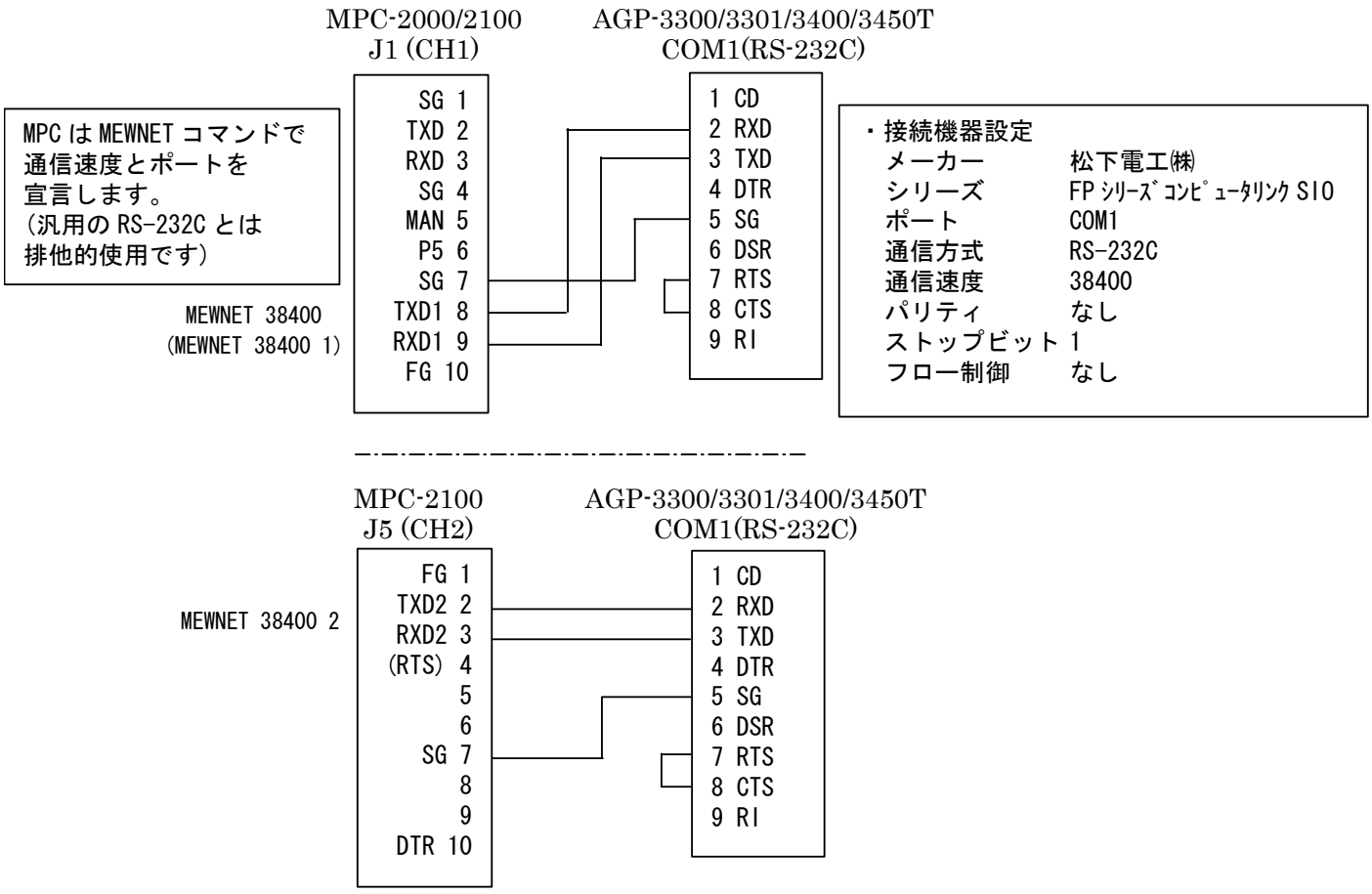
・ プログラム例

```
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
```

◆ デジタル AGP-3300 で作成したティーチング画面の例です。
プログラムは上記と同様です。



◆ デジタル AGP シリーズとの接続例



◆ タッチパネルで実行中の文番号をモニタする

データエリア MBK(7868)～(7899)には各タスクの実行中の文番号が入っています。
タッチパネルのデータ部品(データ表示器)の DT アドレスを対応させて配置すれば各タスクの実行中の文番号を知ることができます。

1) 文番号 65535 以下
・タッチパネルデータ形式 ： DEC(1W 符号無し)

タスク	MBK(),DT	タスク	MBK(),DT	タスク	MBK(),DT	タスク	MBK(),DT
0	7868	8	7876	16	7884	24	7892
1	7869	9	7877	17	7885	25	7893
2	7870	10	7878	18	7886	26	7894
3	7871	11	7879	19	7887	27	7895
4	7872	12	7880	20	7888	28	7896
5	7873	13	7881	21	7889	29	7897
6	7874	14	7882	22	7890	30	7898
7	7875	15	7883	23	7891	31	7899

2) 65535 を越える文番号には S_MBK LONG_PRG を指定します。その時の MBK アドレスは MBK(7836) から 2 ワード毎になります。
・タッチパネルデータ形式 ： DEC(2W 符号無し)
・MPC プログラム： S_MBK LONG_PRG を入れる(LONG_PRG は予約定数)

タスク	MBK(),DT	タスク	MBK(),DT	タスク	MBK(),DT	タスク	MBK(),DT
0	7836	8	7852	16	7868	24	7884
1	7838	9	7854	17	7870	25	7886
2	7840	10	7856	18	7872	26	7888
3	7842	11	7858	19	7874	27	7890
4	7844	12	7860	20	7876	28	7892
5	7846	13	7862	21	7878	29	7894
6	7848	14	7864	22	7880	30	7896
7	7850	15	7866	23	7882	31	7898

GT11 の例

◆ タスクモニタ参照用の文番号付プログラムの保存方法

FTMW メニュー F9[File] ＞ 文番号付プログラム保存 (※FTMW Ver6.39g 以降) により、LIST 表示のイメージで、拡張子 「.F2K_LIST」 として保存されます。

・ 保存例(部分)

```
8880  *CUNET_MAIL
8890    DO
8900     ON   2384
8910     WAIT   IN(2016)<>0
8920     OFF   2384
8930     SELECT_CASE   VOID
```

（この例では、 タスク 14 は 8910 で IN(2016) の入力待ちということがわかります）

※文番号付プログラムを MPC へ LOAD することはできません。

MPC-CUnet、USB-CUnet

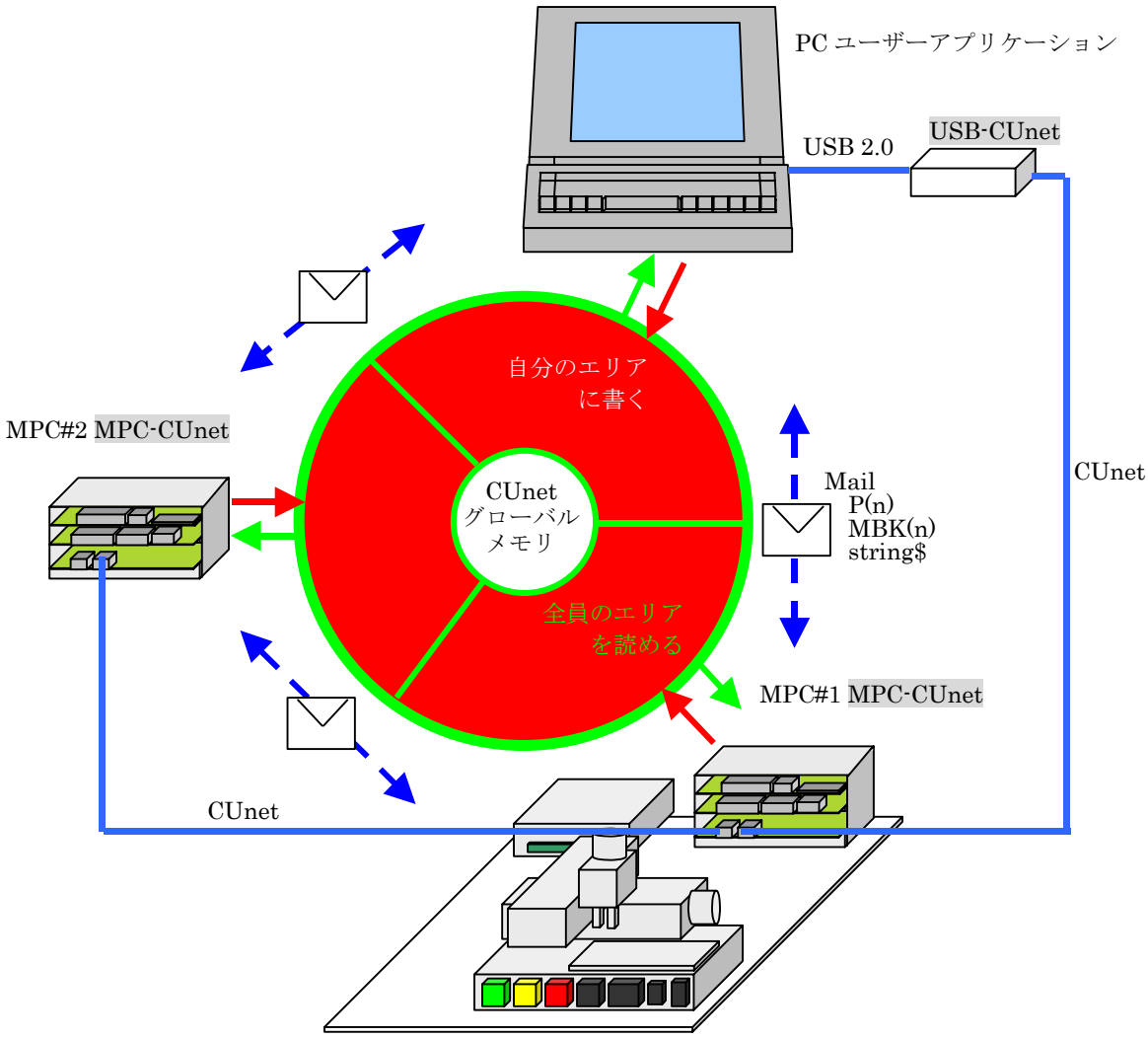
◆ CUnet とは

CUnet は、ステップテクニカ社のオリジナル通信方式であり、シリアル通信回線で接続された複数の拠点間で、同一メモリ空間をリアルタイムで共有するリモートメモリ共有技術です。
MPC-CUnet、USB-CUnet は CUnet デバイス「MKY40」を搭載した通信システムで、複数の MPC 間、MPC～PC 間のメモリ共有を実現、512byte のグローバルメモリを I/O と同様に扱えます。
また、CUnet にはグローバルメモリとは別にメール機能があり、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が行えます。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。
Windows PC の専用 DLL で 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)。

グローバルメモリ読み書きとメールは一緒に使えるので、大量のデータの送受信はメールで、インターロックはメモリで、という使い方も出来ます。

◆ 主なコマンド

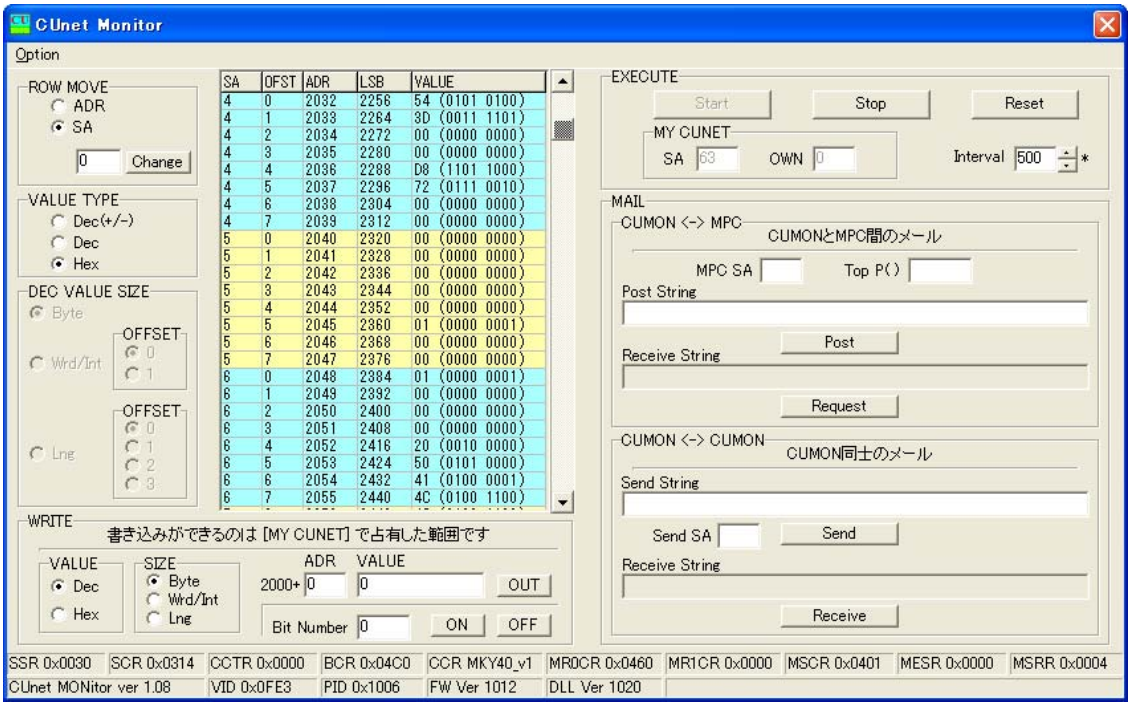
CUNET	MPC-CUnet 初期化
SW、ON、OFF	ビット 2000～6095 操作
IN、OUT	バンク 2000～2511 操作
CU_POST	CUnet メールサーバタスク起動



◆ モニタツール

CUnet モニタ(CUMON.EXE)は CUNET 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。

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。

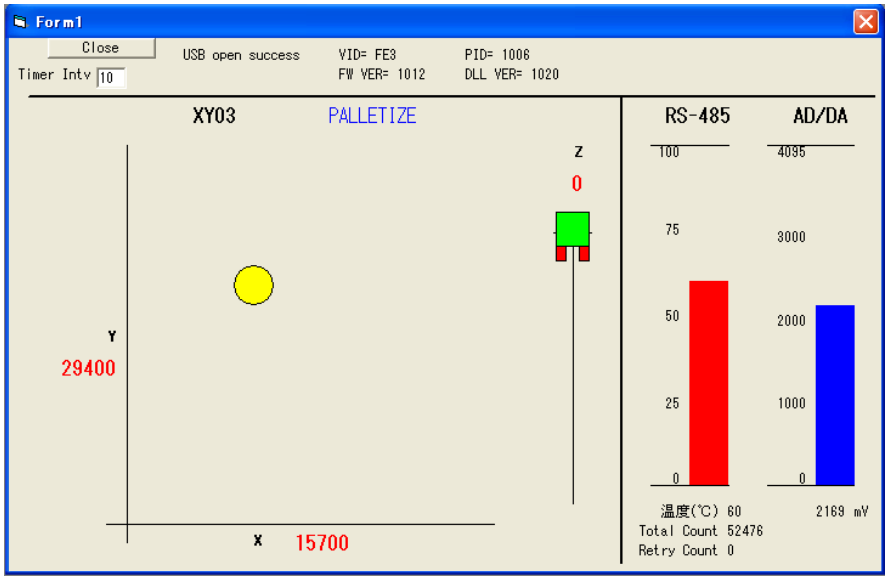


(see also: DOWNLOAD > TOOL ＞ CUnetMonitor)

◆ VB アプリケーション

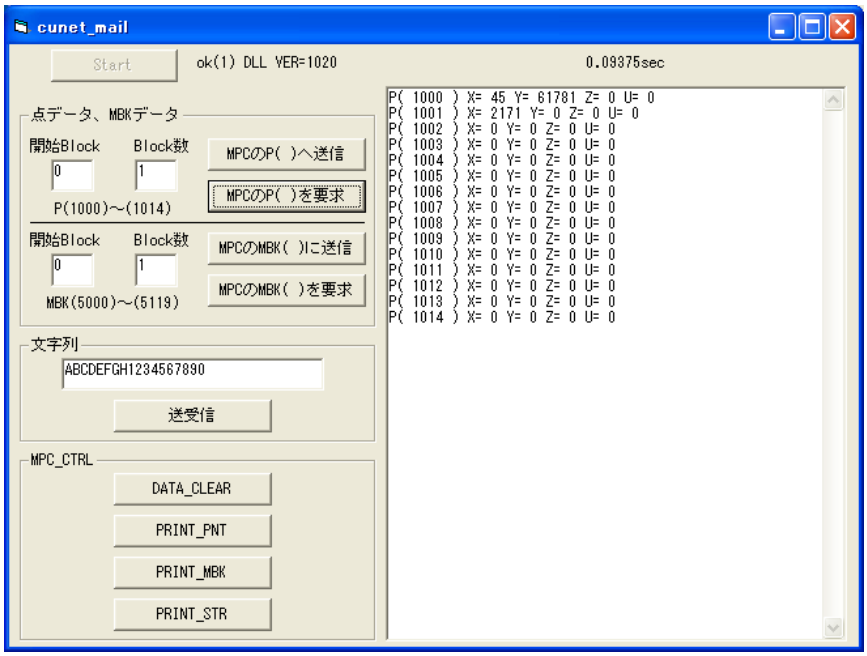
・ グローバルメモリ読み書き

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。



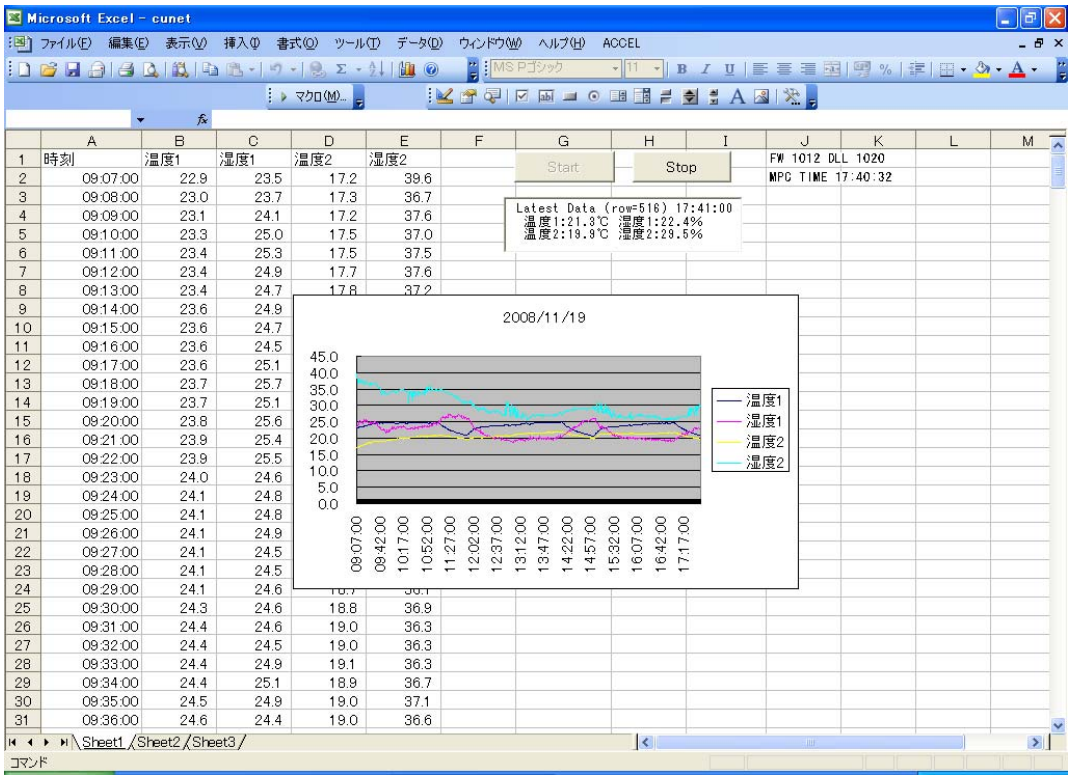
・ CUnet Mail 通信

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。



(see also: アプリケーションノート an2k-002, an2k-003)

◆ タスクモニタの例

USB-CUnet と MPC-CUnet を使って、前述の「タッチパネル接続」と同じように各タスクの実行文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_req_mbk 関数で、MPC の MBK エリアを読み込み MSFlexGrid に入れています。

デバッグ、メンテナンスに利用できます。

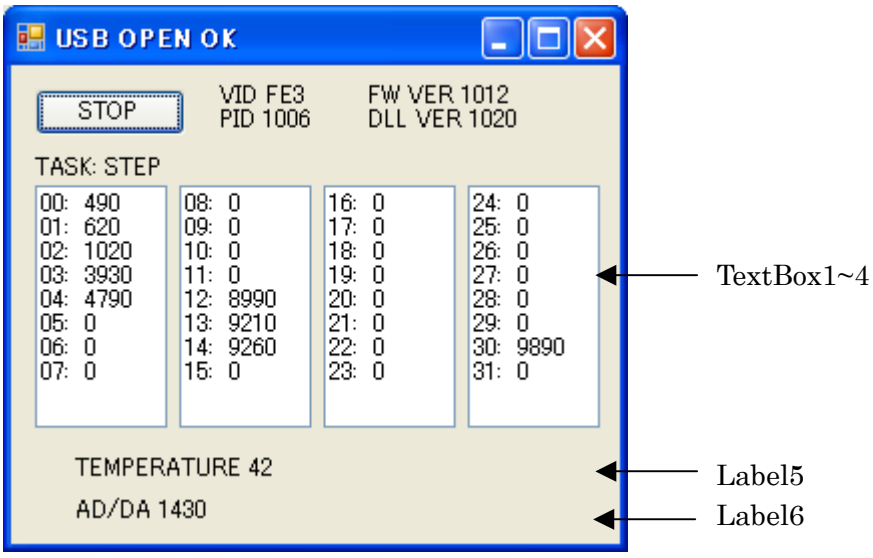
0	0	8	0	16	0	24	0
1	620	9	0	17	0	25	0
2	1020	10	0	18	0	26	0
3	3930	11	3850	19	0	27	0
4	4790	12	8650	20	0	28	0
5	0	13	8870	21	0	29	0
6	0	14	8920	22	0	30	9550
7	0	15	0	23	0	31	0

・ 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 ワード) 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)

USBメモリ

◆ MRS-MCOM は USB メモリ専用ポートを備えており、パソコンで作成した点データやプログラムを読み込むことができます。機種切り替え時などのデータ入れ替えに応用できます。
USB メモリからプログラム、点データのアップデートが行えます(リカバリ機能)。
※USB メモリによってはアクセスできないものがあります。メモリ動作確認結果、リカバリ機能等詳細は MRS-MCOM のマニュアルをご覧下さい。

◆ 主なコマンド

DIR	USB メモリ(MRS-MCOM)のファイルリスト取得
USB_LOAD	USB メモリからプログラム読み込む
USB_SAVE	USB メモリへプログラムを書き込む
USB_PLOAD	USB メモリから点データを読み込む
USB_PSAVE	USB メモリへ点データを書き込む
USB_WRITE	USB メモリファイルへ追記する

◆ プログラムの読み書き
プログラムの更新、保存などメンテナンスに応用できます。

・ FTMW ダイレクトコマンド実行例

```
#LIST 0                                                    /* 現在の MPC のプログラム
10        DO
20         FOR   I=0 TO 2
30          ON   I
40          TIME   100
50          OFF   I
60          TIME   100
70         NEXT   I
80        LOOP
#DIR                                                       /* USB メモリ内容 = 空っぽ
 ドライブ A のボリュームラベルはありません

ファイルが見つかりません
          0 個のファイル                 0 バイト
          0 個のディレクトリ
A:>
#USB_SAVE "TEST.F2K"                                       /* MPC のプログラムを USB メモリに書き込む
#DIR
 ドライブ A のボリュームラベルはありません

2009/00/02  10:46               108 TEST.F2K               /* 新規ファイル
          1 個のファイル                 108 バイト
          0 個のディレクトリ
A:>
#NEW                                                       /* 現在の MPC のプログラムを消去
#LIST                                                      /* 確認表示 = 空っぽ

#USB_LOAD "TEST.F2K"                                       /* さっき書き込んだファイルを読んでみる
#LIST                                                      /* 確認表示
10        DO
20         FOR   I=0 TO 2
30          ON   I
40          TIME   100
50          OFF   I
60          TIME   100
70         NEXT   I
80        LOOP
#
```

「/*コメント」付きのファイルは LOAD できません。(基本的には FTMW で保存されたプログラムが対象です)

◆ 点データの読み書き

機種切り替え、作業データの保存などに応用できます。

・ FTMW ダイレクトコマンド実行例

```
#DIR
 ドライブ A のボリュームラベルはありません

ファイルが見つかりません
           0 個のファイル                 0 バイト        /* 空っぽの USB メモリ
           0 個のディレクトリ
A:>
#USB_PSAVE "TEST.P2K"                                    /* 現在の MPC の点データを保存
#DIR
 ドライブ A のボリュームラベルはありません

2009/03/12  00:28                395 TEST.P2K            /* 新規ファイル
           1 個のファイル                 395 バイト
           0 個のディレクトリ
A:>
#NEWP                                                    /* 現在の MPC の点データを消去
#PLS 0                                                   /* 確認表示
P(1)     X= 0  Y= 0  U= 0  Z= 0
P(2)     X= 0  Y= 0  U= 0  Z= 0
P(3)     X= 0  Y= 0  U= 0  Z= 0
P(4)     X= 0  Y= 0  U= 0  Z= 0
P(5)     X= 0  Y= 0  U= 0  Z= 0
P(6)     X= 0  Y= 0  U= 0  Z= 0
P(7)     X= 0  Y= 0  U= 0  Z= 0
  (略)
#USB_PLOAD "TEST.P2K"                                    /* さっき USB メモリに保存した点データを読込んでみる
#PLS 0                                                   /* 確認表示
P(1)     X= 3440  Y= 17480  U= 0  Z= -19027
P(2)     X= 16420  Y= 18120  U= 0  Z= -18707
P(3)     X= 3200  Y= 43640  U= 0  Z= -19267
P(4)     X= 29100  Y= 17960  U= 0  Z= -18947
P(5)     X= 42020  Y= 17880  U= 0  Z= -18867
P(6)     X= 28920  Y= 43480  U= 0  Z= -19187
P(7)     X= 0  Y= 0  U= 0  Z= 0
  (略)
#
```

◆　ファイルへのデータ書込み

USB メモリのファイルにテキストデータを書込みます。

USB_WRITE コマンドは APPEN OPEN -> WRITE -> CLOSE を一連で行います。

データロガーとして応用できます。

・ FTMW ダイレクトコマンド実行例

```
#LIST                                                  /* 現在 MPC に入っているプログラム表示
10       FILE$="TEST.CSV"                              /* ファイル名。FILE$ は予約変数
20        USB_DEL   FILE$                              /* 既存の同名ファイル消去
30        DO
40          FORMAT   "00/00/00"
50       DT$=HEX$(DATE(0))
60          FORMAT   "00:00:00"
70       TM$=HEX$(TIME(0))
80          USB_WRITE   DT$+","+TM$+"¥n"               /* USB 書込み
90          TIME   1000
100       LOOP
#DIR                                                   /* USB メモリ内容 = 空っぽ
 ドライブ A のボリュームラベルはありません

ファイルが見つかりません
           0 個のファイル                 0 バイト
           0 個のディレクトリ
A:>
#RUN                                                   /* 実行

@None_file                                              /* 既存ファイルがない場合のメッセージ(停止はしな
い)
       *0    [90]                                      /* しばらくして Ctrl+A で停止。
#DIR                                                   /* USB メモリ内容
 ドライブ A のボリュームラベルはありません

2009/00/02  11:02                133 TEST.CSV        /* 新規ファイル
           1 個のファイル                 133 バイト
           0 個のディレクトリ
A:>
#TYPE "TEST.CSV"                                       /* ファイル内容を表示
09/00/02,11:02:17
09/00/02,11:02:18
09/00/02,11:02:19
09/00/02,11:02:20
09/00/02,11:02:21
09/00/02,11:02:22
09/00/02,11:02:23

A:>
#
```

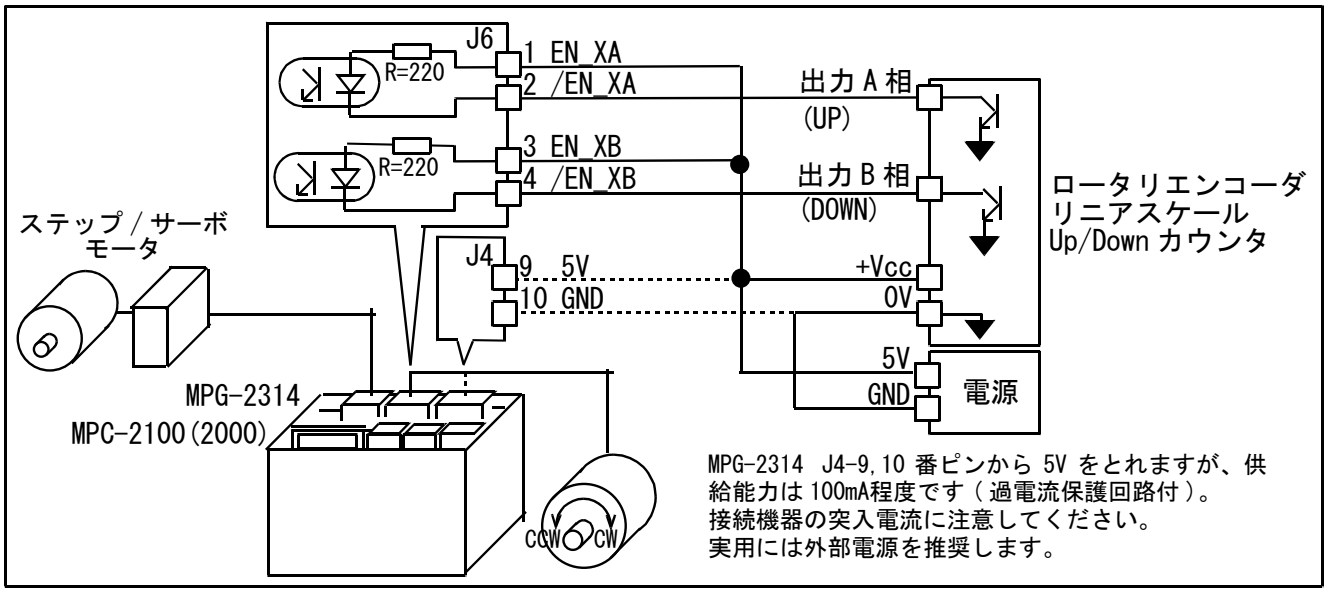
エンコーダ、カウンタ入力

◆ 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)

AD/DA変換、PWM制御

◆ MPC-AD12 は 12bit AD/DA 変換ボードです。

AD 入力・DA 出力レンジ 0～4.095V (入力レンジは IC 交換で±10V または 0~2.5V に変更できます)

AD : 8 チャンネル

DA : 4 チャンネル

◆ 主なコマンド

AD	アナログ電圧入力値
DA	アナログ電圧出力設定
SET_AD	AD コンバータのタイプや平均値サンプリングのサンプル個数を指定
PWM	出力ポートを PWM 制御

◆ サンプルプログラム

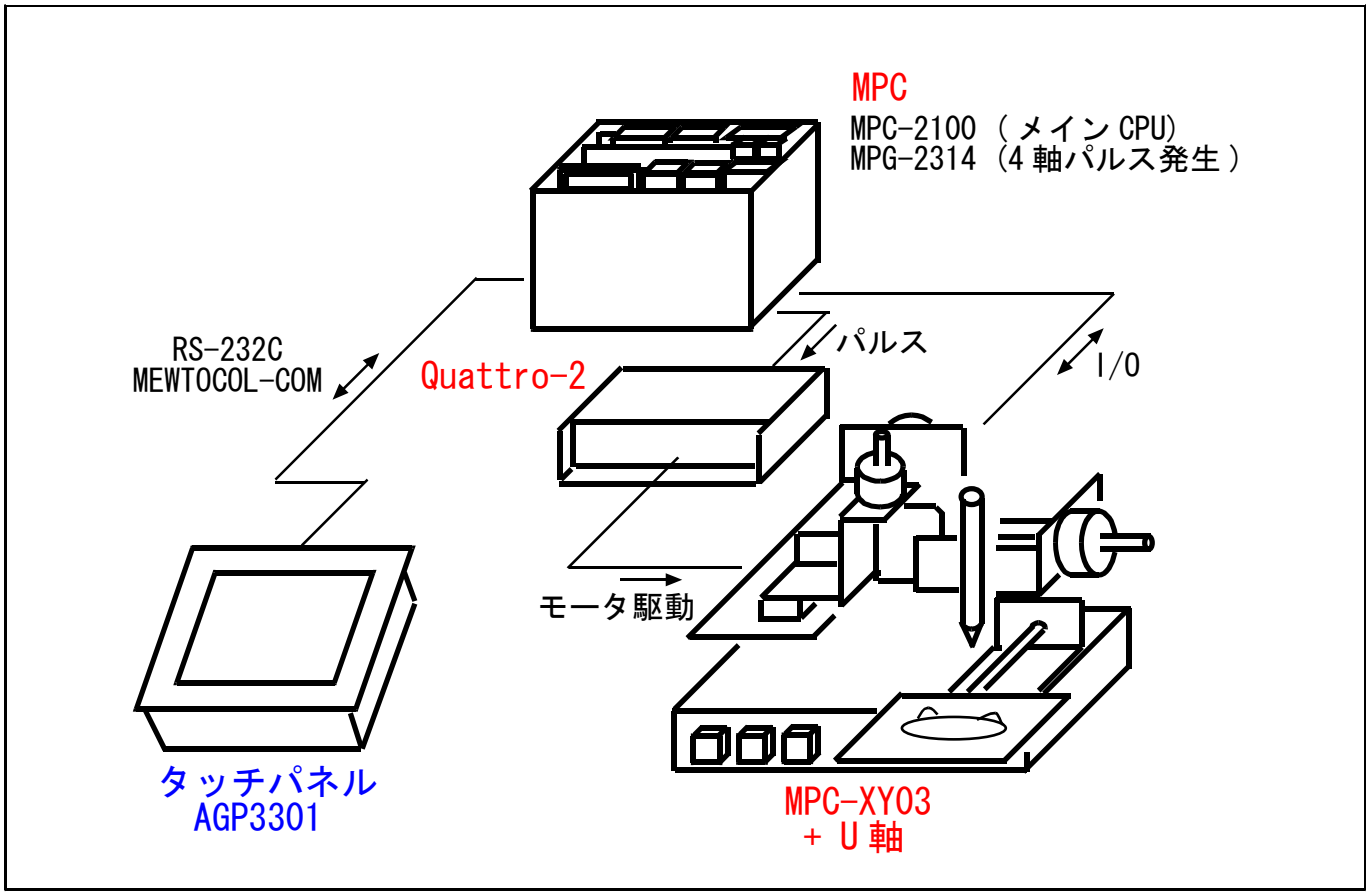
```
A_=AD(0)                    /* MPC-AD12 ch0 アナログ入力
DA  A_ 0                    /* MPC-AD12 ch0 アナログ出力
PWM 12 A_/100               /* 出力ポート 12 PWM 制御
```

MPC-XY03デモ機の実例

MPC-2000/2100

直線・円弧・連続補間

■機器構成



■概要

- 直線と円弧の連続補間機能を使って絵や文字を書きます。
- 塗布機や切削機などに応用できます。

■特徴

直線補間、円弧補間、連続補間

- MPG-2314 は 4 軸パルス発生ボードです。最大 8 枚まで拡張できます。
- 任意の軸の組み合わせで直線補間(3 軸まで)、円弧補間(2 軸まで)が行えます。
- 直線と円弧の補間ドライブを連続して行うことができます。
- エンコーダ入力も装備しています(標準 2ch、拡張可)

タッチパネル

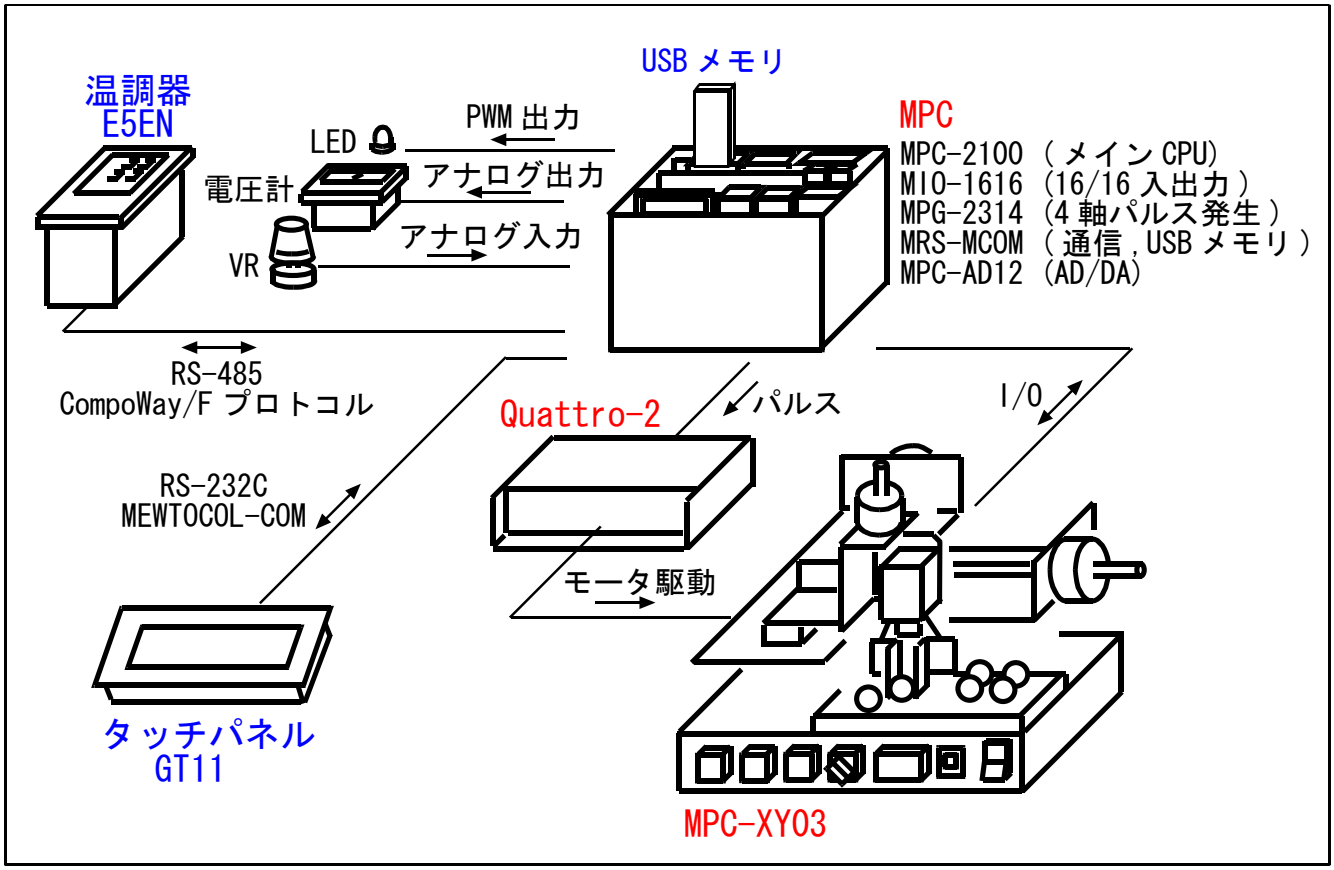
- MPC は松下電工 MEWNET-FP シリーズの PC 間双方向通信「MEWTOCOL-COM」に準拠したプロトコル(スレーブ側)を実装し、パナソニック電工㈱ GT シリーズ、㈱デジタル GP シリーズの「ダイレクトアクセス方式」に対応しています。

Quattro-2

- 4 軸を 1 つにパッケージしたステップモータドライバです。
- 2 相ステッピングモータ ユニポーラ駆動。
- 軸毎にマイクロステップ 1/2、1/4、1/8、1/16 分割、ドライブ電流 0.1～3A を設定可能。

MPC-2000/2100
ワーク搬送、温調、AD/DA

■機器構成



■概要

- 一般的な FA 機器を想定したデモ機です。
- XYZ ロボットのワーク搬送、温調器制御、AD/DA 変換、USB メモリアクセス、タッチパネルユーザーI/F を実装しています。

■特徴

RS-232C/422/485 通信、USB メモリ

- MRS-MCOM で RS-232C/422/485 を拡張できます。(1 枚 3ch、最大 3 枚)
- オムロン CompoWay/F プロトコル(準拠)を搭載しています。
- MRS-MCOM は USB メモリの読み書きもできます。データのログ・切り替え、メンテナンスなどに応用できます。

AD/DA 変換

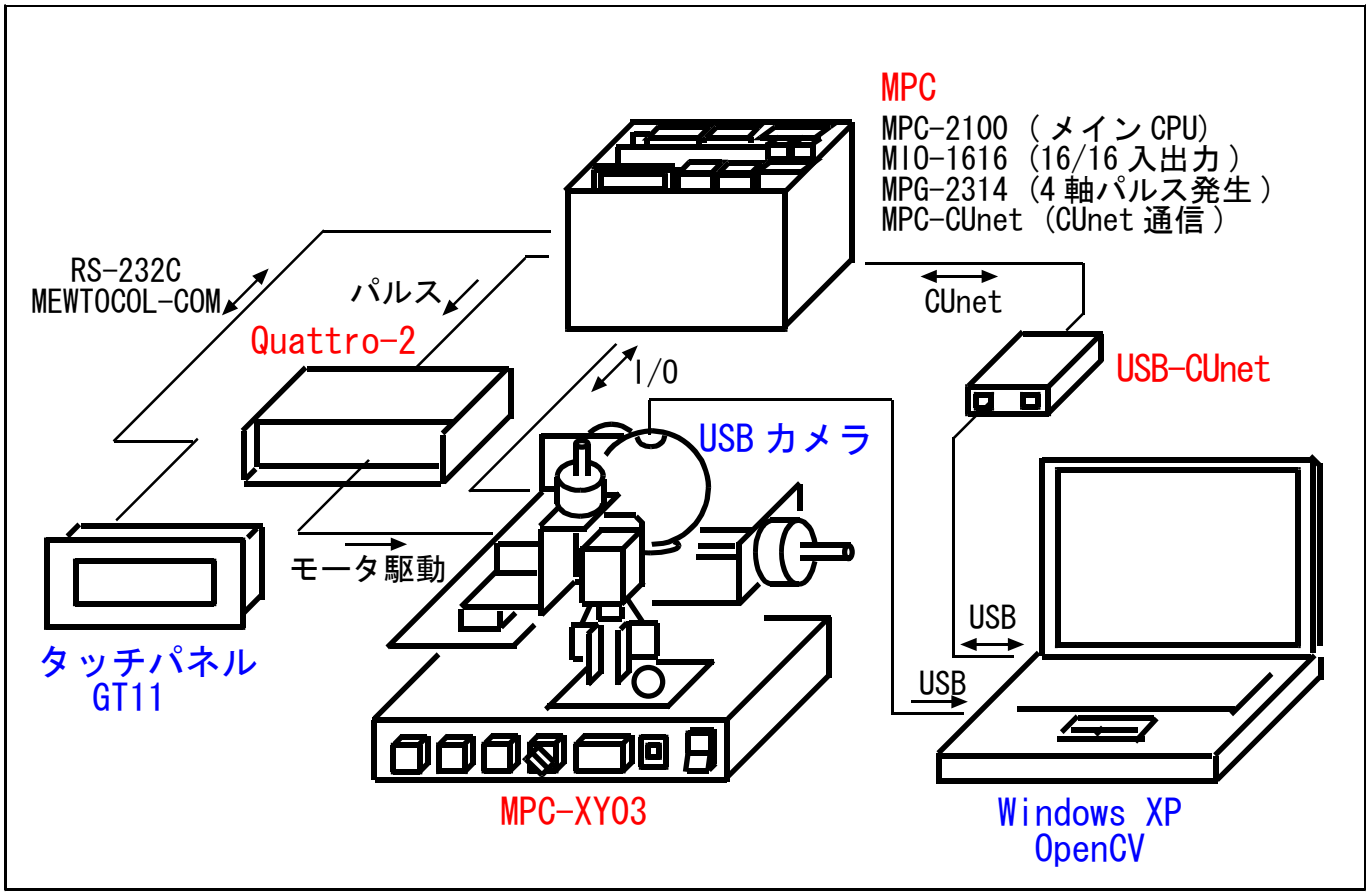
- MPC-AD12 は 12bitAD/DA ボードです。
- 入出力レンジは 0～4.095V で AD:8 チャンネル DA:4 チャンネルです。

CompoWay/F プロトコルとは

- CompoWay/F プロトコルとはオムロン製コンポーネント機器でサポートしている、シリアル通信用のオムロン専用プロトコル。「miniFINS」ともいう。（出典：オムロンホームページ FA 用語辞典）

MPC-2000/2100
画像処理

■機器構成



■概要

- 画像処理搭載装置を想定したデモ機です。高価な画像装置の代わりにノートパソコン、USB カメラとフリーのソフトウェア「OpenCV」を用いています。
- パソコンで球体ワークの中心座標(画像の XY ピクセル座標値)を検出し、それを USB-CUnet 経由で MPC に伝えます。MPC は受け取ったデータをメカのパルス座標に変換しワークをピックアップします。

■特徴

USB-CUnet

- パソコンと MPC 間でメモリ共有とデータ転送が行えます。
- USB-CUnet は他社の CUnet(MKY40)機器とも接続できます。

CUnet とは

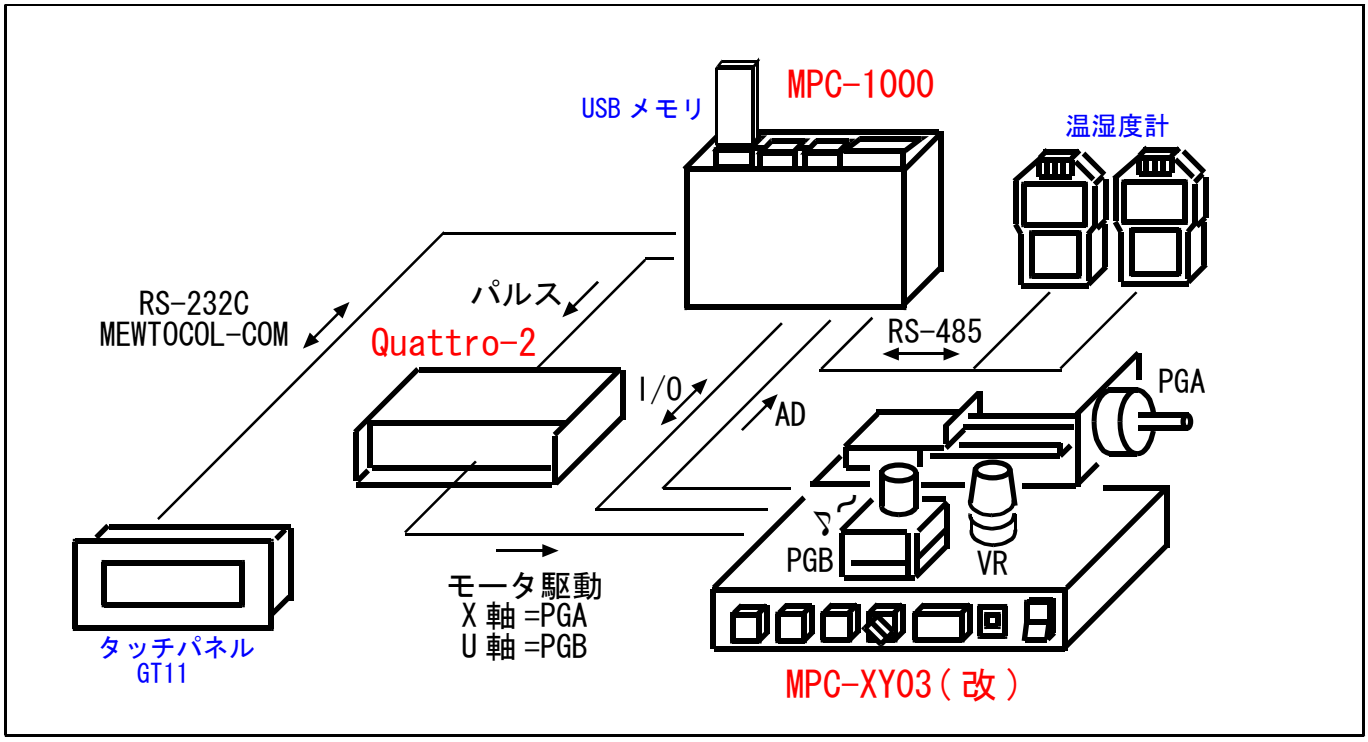
- CUnet は、㈱ステップテクニカ社のオリジナル通信方式であり、シリアル通信回線で接続された複数の拠点間で、同一メモリ空間をリアルタイムで共有するリモートメモリ共有技術です。

OpenCV とは

- OpenCV（おーぷんしーぶい）とはインテルが開発・公開したオープンソースのコンピュータビジョン向けライブラリ。 現在は Willow Garage（うぃろー・がれーじ）が開発・サポートを行っている。(出典: フリー百科事典『ウィキペディア（Wikipedia)』)

MPC-1000
小型卓上機（ステップモータオルゴール）

■機器構成



■概要

- 少数 I/O とステップモータで構成された小型卓上機を想定したデモ機です。
- ユーザーインターフェースはタッチパネルです。
- パルスモータテーブルのピッチ送り(位置制御)、定速回転、USB メモリからファイルを読み込んで動作変更、などを行います。
- USB メモリに保存されている点データのフォーマットに従い PGB のパルスレートを変えて音階を作り、オルゴールのようにメロディーを演奏します。
- タッチパネルの鍵盤操作を USB メモリに保存、再生します。再生スピードは VR(AD 入力)で調整します。
- タッチパネルの時計で MPC-1000 の時計をセットします。
- RS-485 でマルチドロップ接続した温湿度計のデータを USB メモリに CSV 形式で記録します。

■特徴

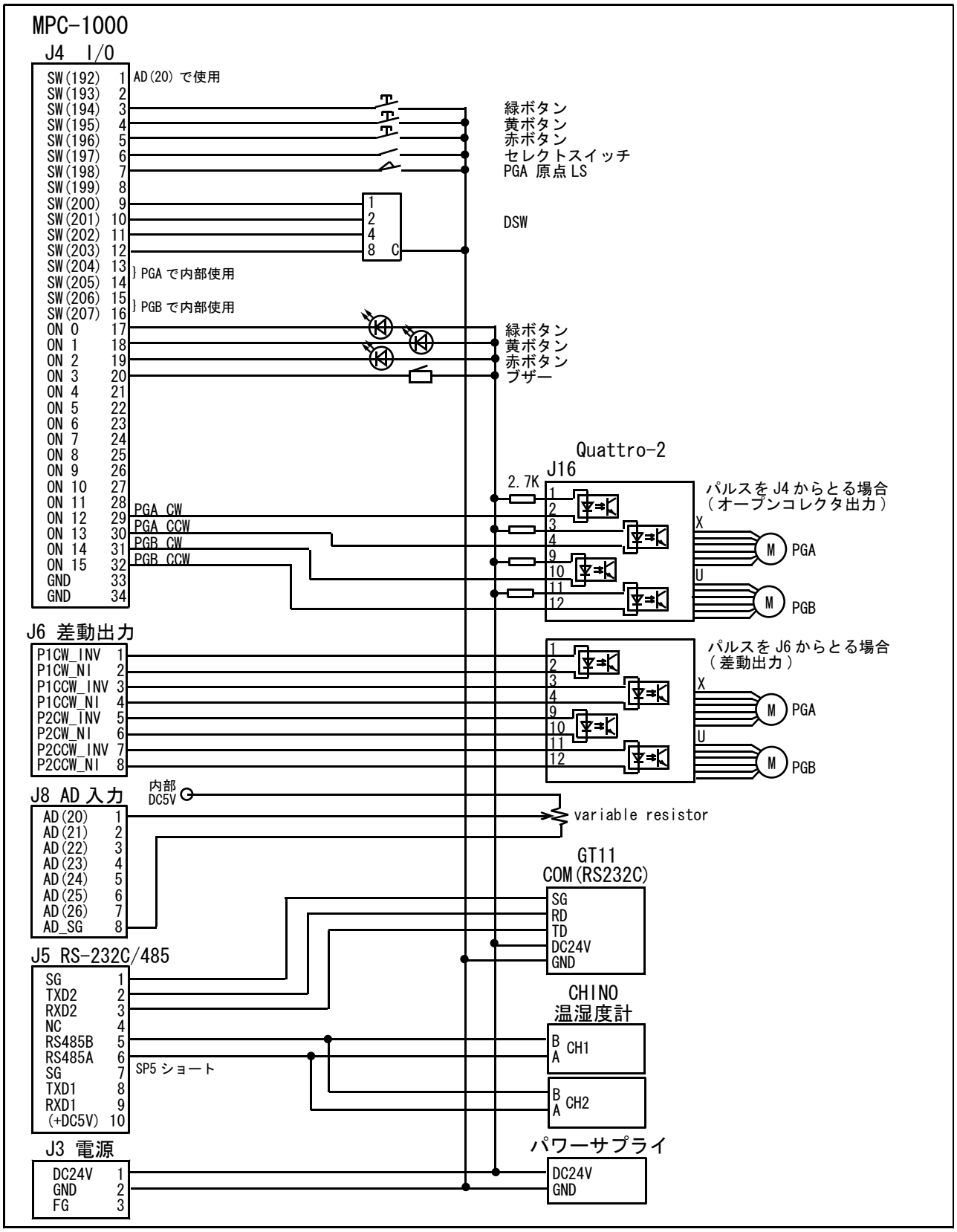
MPC-1000

- 入力 16 点(そのうち 7 点は AD と兼用、パルス使用時は 1 軸につき 2 点が PG 制御用となります)
- 出力 16 点(パルス使用時は 1 軸につき 2 点がパルスポートになります。差動パルス出力と排他使用。)
- RS-232C 3ch(プログラム 1、ユーザー2、そのうち 1ch は RS-485 と排他使用)。パナソニック MEWNET-COM、オムロン CompoWay/F 準拠プロトコル搭載。
- USB メモリを利用してデータロガーなどにも応用できます。
- 言語仕様、拡張ボード・ラック、プログラム開発環境は MPC-2000 と共通です。
- 詳細はパンフレット、マニュアルでご覧下さい。

Quattro-2

- Quattro-2 は 4 軸の 2 相マイクロステップモータドライバですが、本機ではそのうち 2 軸を異なる設定で使っています。設定は専用ソフトで行います。
  X 軸(PGA)：励磁モード 1/4 分割、ドライブ電流 0.6A、スリープ電流 0.1A
  U 軸(PGB)：励磁モード 1/2 分割、ドライブ電流 0.8A、スリープ電流 0.4A

■結線



- RS-485 と RS-232C CH1 は排他的使用。
- USB-RS2 を用いてデバッグする場合、USB-RS2 の SP3-SP4 のショートジャンパーを抜いてください(出荷状態では SP3-SP4 がショートされており、J1 の TXD1 が RXD1 にループバックしてしまいます)。

＊＊＊＊＊＊＊＊

ACCEL
http://www.accelmpc.co.jp
tech-sp@accelmpc.co.jp

＊＊＊＊＊＊＊＊