目次

第 1 章　MPC-3000 導入ガイド

1-1 バージョンについて

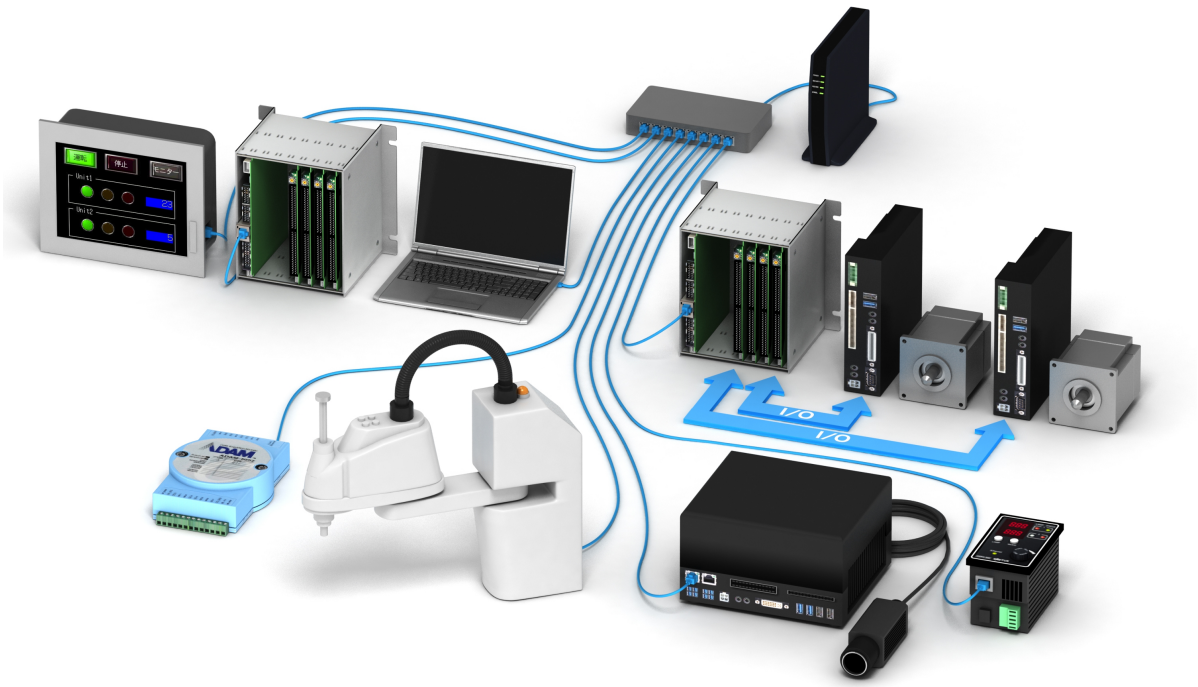
このマニュアル、以下のバーはジョン以降を対象としています

- MPC-3000 BL/I 2.01_30 2019/07/10
- MPC_Monitor_Telnet 1.03
- MPC_Monitor 1.73
- FTMW2K 1.17

1-2 ハードウェア

CPU

MPC-3000 は MPC-2000 シリーズの後継機種として、イーサネット(TCP/IP)を備えたボードコントローラです。CPU にはルネサス SH7216(200Mhz)を採用し、従来製品比 1.5 倍の高速性能を備えています。電源は DC24V 単一で、市販の電源を使用できます。ボード単体の消費電流は 200mA 程度で、軽負荷、低発熱で動作します。ラックは 2,4,8,16 枚用と 4 種類用意されており、様々な周辺ボードと併せてシステム構築することができます。



周辺ボード

MPC-2000 シリーズで用意された、I/O ボード、パルス発生ボード、拡張 RS-232(422,485 対応)、CUnet,AD/DA ボードの全てにコマンド互換で対応します。
このため、すでに稼働中の装置の CPU を MPC-3000 に入れ替えるだけで、ソフトウェアはそのままにイーサネット対応とすることができます。

1-3 MPC-2000 との相違点

MPC-2000 から MPC-3000 に移行する場合の注意点が 2 つあります。

1 つは、RUN の意味が異なっていることです。MPC-2000 シリーズでは最初の応答タスクが 0 で、RUN は、この TASK 0 が主体となって、プログラムを実行しましたが、MPC-3000 では、最初の応答タスクは 47 となっており、RUN は、プログラムを TASK 0 で別タスクとして実行することを意味します。
このため、プログラム実行後もコマンドが有効となります。プログラム実行中も実行状態の確認、監視が可能です。

ただし弊害もあり、この場合起動されたタスクがプログラムポートからの入力"INPUT"を含む場合、監視 TASK 47 と入力競合することになります。これを避けるには、実行を"RUN -1"とします。これにより、TASK 0 を起動せず、TASK 47 でプログラムが実行され、MPC-2000 シリーズと同様の実行環境となります。

もう 1 つの相違は、タッチパネル通信を設定する MEWNET コマンドが無効であることです。
MPC-3000 では、タッチパネルの設定は後述する SET_IP コマンドで TCP_IP 設定と同様に扱われます。

例えば、CH2 で 38400、8bit でタッチパネル通信を行う場合は、

```
SET_IP Serial MEWNET 2 38400
```

とします。この設定により、タッチパネル通信が TCP/IP 通信と同様バックグラウンドで行われます。また指定できる RS ポート番号には制限がなくなり、1～14 の全てのポートを割り当てる事ができます。

MBK エリアについて

PLC のメモリシェアでは、一般にワードのデータ領域は、D 或いは DT と表現されています。MPC でこの領域に相当するのは、MBK()エリアです。これに関連するコマンド関数は、S_MBK,MBK()などですが、MPC-3000 では互換性の観点から MBK を DT と記述できるようになっています。S_MBK を S_DT,MBK()を DT()と記述しても同じ意味になります。

1-4 情報交換

プログラム転送

TCP/IP Telnet を経由したプログラム転送が高速となりサイズの大きなプログラムの保守、変更に効果を発揮します。以下に約 100kbyte のプログラムの転送の場合の比較を示します。
Telnet を利用したプログラム転送は圧倒的に高速となります。

約 100k のプログラムサイズ	ロード	セーブ
RS-232 38400 bps	38 秒	32 秒
RS-232 115200 bps(FTMW1.17)	15 秒	13 秒
Telnet 経由(*注)	8 秒	1 秒

注) Windows7 の標準状態では、セーブの時間が長くなります。この改善方法については、
4-2 MPC_Monitor_Telnet の⑨の注を参照してください。

ファイル

MPC-3000 では、microMMC カードに対応します。
(MMC とは 2G までのサイズで SPI モードを備えた SD カードのことです)。
USB は、 1 行ごとのログデータの書き込みを想定しています。通信速度が遅い為に大量のデータの読み書きには向いていませんが、稼働時に抜き差しができます。
MMC カードは高速で読み書きが出来ますが、カード挿入位置がボード後ろ側になるために、固定外部メモリとして使用してください。

PC 通信

通信では、CUnet、TCP/IP を用いた情報交換が可能です。CUnet は USB インターフェースによって PC データと情報交換ができます。USB-CUnet 用の DLL ライブラリが用意されています。
TCP 通信については、Telnet,TCP/UDP パケット通信、Modbus-TCP、MEWTOCOL7,MC プロトコルなどが用意されています。TCP 通信は、最大で 7 ポート対応させることができます。
7 ポートは少な目の数ですが、同じ LAN 内での 1 つの装置の TCP/IP 通信の現実的なポート数は、3～4 と考えられています。これは、イーサネットがコリジョン(通信衝突)を前提とした通信であり、実際のスループットが 20%程度という経験則にもあてはまります。
CUnet と TCP/IP の使い分けですが、リアルタイム性を必要とする場合は、CUnet を使用します。メモリシェアでは 512byte という制限はありますが、CUnet では数 msec での情報の同期が可能です。対して、TCP/IP では、より多くのデータのやり取りが可能ですが、応答速度を数 10msec 以下にすることはできませんし、コリジョンによる遅れ(1 秒程度)がランダムに発生します。

シリアル通信

MPC-3000 では、本体で 3 ポート、拡張で 12 ポート計 15 ポートのシリアル通信(RS232)を扱うことができます。ボーレートは、すべてのポートで、115200bps まで設定可能です。

また、CPU の CH2 と、MRS-MCOM6 の拡張ポートでは RS-485 に対応し、Modbus-RTU 通信にも使用できます。

現在では、TCP/IP による通信が普及してきていますが、小規模のデータ通信では、前節で述べたコリジョンの影響や、イーサネット通信の冗長性により、RS-232 のデータ通信の方が有利な場合が多くあります。例えば、バーコードのようなデータで 20byte 未満のキャラクタ通信では、RS-232 の方が応答は高速になります。

1-5 プログラム

プログラムは、Basic ライクなインタプリタによって行います。通常の Basic と異なりマルチタスクの機能があり、40 タスク程度の異なったプログラムを 1 つの CPU で実行することができます。これにより、複雑で大規模な装置のプログラムをユニット毎に区別して記述することができます。

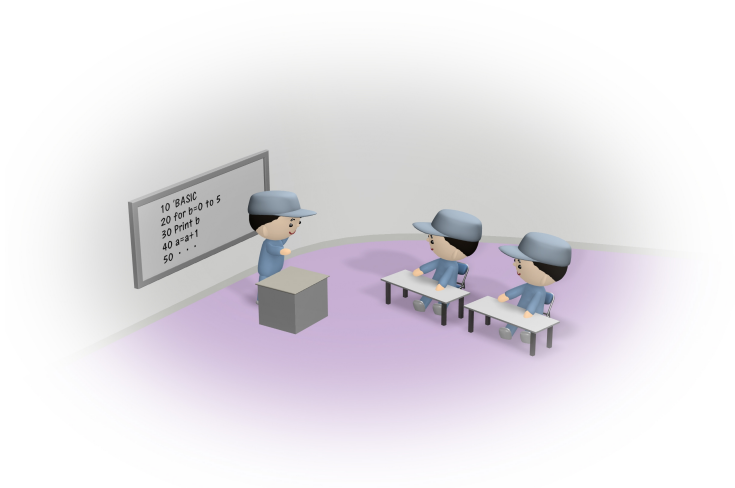
また、インタプリタであるために、加筆、削除後のコンパイルや転送といった手続きが不要で、ただちに実行することができるため、プログラムのカット&トライが容易です。

1-6 MPC-3000 を使用する利点

言語は BASIC ライク

MPC-3000 は、PLC のラダー回路と異なり言語ソフトによって動作するため、数値データを直接扱うことができ、分岐やリトライなどの記述も容易となります。さらにプログラムの規模と動作速度が無関係であるため、複雑な動作を記述しても装置の動作速度が遅くなることはありません。

また、MPC-3000 では、TCP/IP 通信もパケット通信による文字列処理として扱えるため、PC ソフトのように専門的な知識やノウハウを必要とすることなく、インタプリタ上で簡単に実現することができます。



MPC の言語は、最近の PC で多用される C++や Java 系の言語と異なり、１行のコマンドが装置の動作に直接結び付く言語です。そして１行の基本実行速度は 19μ 秒と高速です。
1msec で 50 行程度実行されるため、装置制御には十分な速さと言えます。

PLC では１行の実行速度は遥かに高速ですが、実際の反応速度となると、すべてのプログラムをスキャンする速度となるため、実際には MPC の 19μ 秒/行の実行速度より低速になります。

OS

MPC-3000 には純正のマルチタスクモニタが搭載されています。このため、Unix 系 OS のボードコンピュータのように起動に時間がかかったり、装置特有のリアルタイム処理に苦労するということがありません。TCP/IP 通信については、ルネサス社の TCP/IP プロトコルスタックを使用しており、マイコンに負荷のかからない高速な処理を実現しています。また、一般の公開 OS ではありませんのでウィルスの影響をうけることもありません。

第２章　TCP/IP と SET_IP コマンド

MPC-3000 は、TCP/IP 通信機能を備えています。この機能を有効にするには、IP アドレス設定、TCP/IP 通信機能設定が必要です。TCP/IP 通信機能設定は９個まで設定できますが、シリアルインターフェースを用いたタッチパネルプロトコル MEWNET も同じ SET_IP エリアを用いますので、MEWNET 使用分 TCP/IP 設定チャンネルが減ります。
TCP/IP の設定は FTMW2K を用いて、MPC-3000 と RS 接続しコマンドで行います。

2-1 IP アドレスの設定

コマンドは **SET_IP** です。以下に入力例を示します
入力例:

```
SET_IP    192 168 0 14  255 255 255 0  192 168 0 248
          { Ip adress }  {  ネットマスク  }  {  Gate way  }
```

IP アドレスのみ変更の場合は、**"SET_IP 192 168 0 6"**のように必要なところまで入力します。この設定値が有効になるのは、パワーオンリセット以後、もしくは**"RESET"**コマンド実行後です。また、MASK,GATEWAY をそれぞれ独立して入力することができます。

MASK の入力例：

```
SET_IP IP_MASK 255 255 255 0
```

GATEWAY の入力例：

```
SET_IP IP_GATE 192 168 0 248
```

2-2 Delay ACK

Delay ACK は、PC 間の大容量データ通信で用いられる手法です。MPC では、Delay ACK は使わない設定としています。機器によって対応がわかれるためです。万一、Delay ACK が ON と表示されたら、SET_IP ND_ACK を入力し OFF としてください。

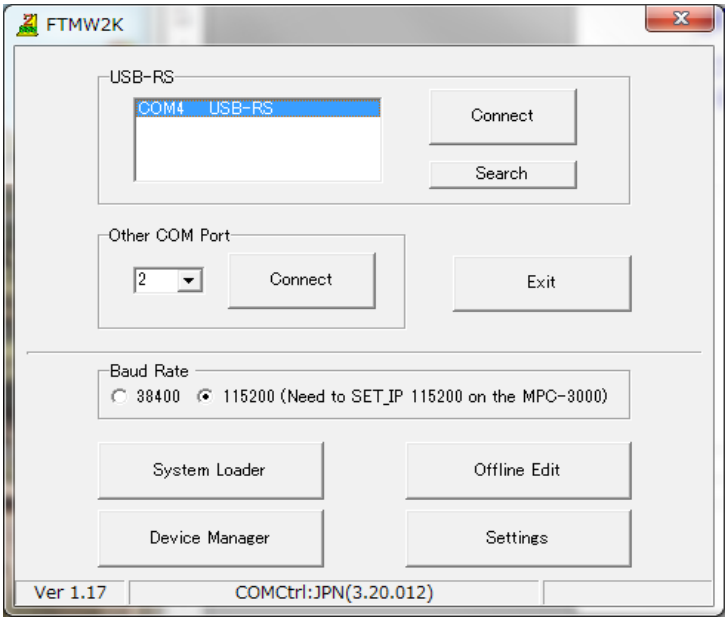
SET_IP ND_ACK	標準設定
SET_IP D_ACK	Delay Ack 有効(非推奨)

2-3 ボーレート

TCP/IP の設定とは無関係ですが、**SET_IP** コマンドでプログラムポートのボーレートを 38400bps もしくは 115200bps に設定することができます。

SET_IP 38400	標準設定
SET_IP 115200	115200bps 対応のプログラミングツールで使用します

対応バージョンのオープニング画面では以下のように、Baud Rate の選択エリアが表示され、115200bps を選択することができます。



設定をし終えたら一度パワーオンリセットします。
上記の設定場合、表示は以下のようになります。

```
MKY =0
++++++++++++++++++
IPadrs = 192.168.0.14
 MASK  = 255.255.255.0
Gateway= 192.168.0.248
OFF Ðelay ACK  baud=38400
MACadrs= 70B3Ð553Ð00F
Ether Opened
TCP_STACK_RAM=11768
TCP Init Completed
Wake TCP
Wake UÐP
Wake Serial
USB_I/F  Ver1.12
iMPC-3000
#
```

2-4 TCP/IP の設定

MPC-3000 にはあらかじめ備えられた TCP/IP 通信機能があります。現在以下の機能が装備されています。Packet 通信は、TCP/IP の通信を文字列で行うことができ、無手順のメッセージ交換や MC_Protocol の Q3E フレーム、Modbus-TCP フレームを生成することができます。なお、すべて

の TCP/IP 設定を消去するには、**"SET_IP　VOIÐ"**を実行します。

機能	用途	種別	補記
Telnet(Server)	メンテナンス	TCP	MPC_Monitor_Telnet 使用
Modbus(Server)	タッチパネル・PC	TCP	Header が 0x50
Modbus(Master)	メモリシェア	TCP	SRC,ADAM オプション
Mcprotocol(server)	タッチパネル・ＰＣ	TCP/UDP	Q3E フレームに対応
Mewtocol7(server)	タッチパネル	UDP	
SHARE	メモリシェア	UDP	MBK エリアの共有
Packet 通信(双方向) Q3E Modbus	汎用	TCP/UDP	MC_protocol(Header が 0x50) Modbus 通信可

Telnet

設定: **SET_IP　TCP Telnet**

Telnet を設定するとパワーオンリセット後、Telnet ポート(23)が開かれ、MPC_Monitor_Telnet によって、プログラムのセーブロード、実行、デバッグなどの作業を行うことができます。Telnet 接続が成立すると、RS-232 ポートによる FTMW2K の接続は無効になりますが、FTMW2K 側で CTRL_A を押すと、RS-232 側に制御が戻ります。

Telnet を使用する利点は、プログラムポートへの RS コネクタの接続が不要となり、LAN に接続した PC からプログラムメンテナンスを行うことができることです。

また、ルータの適切な設定により、インターネットからの、リモートメンテナンスも可能になります。この場合、ルータのアドレス変換では使用ポートの変更も行ってください。

Telnet の標準ポートである 23 をインターネットに向かって開くと、多くの外部アクセスにさらされます。これは、ハッカーやボットと呼ばれる不正アクセス専門のプログラムによるアクセスです。

アクセスによっては、MPC-3000 の実行状態を不安定にしてしまうものもありますので、インターネットからの Telnet はルータで 23 以外のポート番号から、23 ポートに変換して使用します。

MEWTOCOL7

設定: **SET_IP UÐP MEWTOCOL7　ポート番号**

MEWTOCOL7 は、Panasonic PLC 用のメモリリンクプロトコルです。MPC-3000 では UDP 通信のタッチパネルインターフェースとして使用することができます。NEWTOCOL7 には独自の CRC が付加されていますので、UDP 通信でも信頼性の高い情報交換が可能です。ポート番号を指定して設定すると、タッチパネルからの呼びかけに応答します。ポート番号は任意ですが、255 以下を指定する場合は、以下の例にならってください。

SET_IP UÐP MEWTOCOL7　I_PORT¦240

MPC-3000 とタッチパネルのデータエリアの対応は以下のようになります。

MPC-3000	タッチパネル	補記
MBK(0)～MBK(7899)	DT00000～DT7899	MBK()は DT()とも記述可
70000～79900	R0000～R0990	MBK(7900)～MBK(7999)

他に MPC-3000 の I/O エリア、メモリ I/O、CUnet エリア、点データエリア、各種変数等、ほとんどのリソースをタッチパネルで参照変更できます。これについては別途技術資料を参照してください。また、タッチパネルでは送信ディレイのデフォルトが 0 となっている場合が多いようです。送信ディレイは 20msec 以上、受信タイムアウトは 1 秒以下が推奨です。

MC_PROTOCOL

設定 : **SET_IP　　TCP MC_PROTOCOL ポート番号**

この設定は、受動(Slave)MC プロトコルをポート番号に従って開きます。応答ルールは以下のとおりです。

種類	エリア指定	MPC
一括設定　word	Y (0x9d)	OUT に相当
	D (0xa8)	MBK()
	R (0xaf)	MBK(7900)~
一括設定　bit	B (0xa0)	CUnet エリア
	Y (0x9d)	ON/OFF
	D (0xa8)	MBK()
	M (0x90)	ON,OFF -1 ～
on 読み取り word	D (0xa8)	MBK()
	R (0xaf)	MBK(7900)~
	W (0xb4)	CUnet エリア
読み取り bit	X (0x9c)	入力ポート SW()
	Y (0x9d)	出力ポート SW()
	M (0x90)	メモリ I/O　SW(-1~)
	B (0xa0)	CUnet エリア

Modbus-TCP

設定: **SET_IP TCP Modbus**

この設定は、受動(Slave)Modbus プロトコル、ポート 502 を開きます。応答ルールは以下の通りです。

MPC-3000 は Modbus アクセスを受け付けます。これにより Modbus 対応のタッチパネルと接続可能です。配置は以下のようになります。この配置は三菱 PLC,Modbus 対応に準拠しています。(Modbus では 0 番のコイル、0 番のレジスタが定義上ないことに注意)

MPC-3000		Modbus		三菱 PLC
I/O エリア	入力 192～	コイル　10001～		X エリア
	出力 0 ～	コイル　00001～		Y エリア
メモリ I/O	-1～	コイル　8193～		M エリア
CUNET I/O		コイル　30721	レジスタ　430721～	B,W エリア
MBK(0)～(DT エリア)	MBK(0) ～		レジスタ　400001～	D エリア
R エリア　70000～		コイル　22529		L エリア

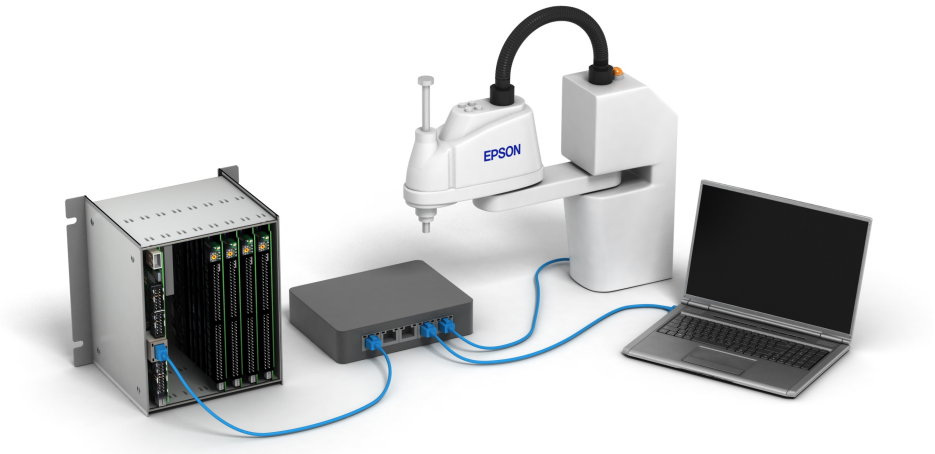
設定: **SET_IP TCP Modbus　port 番号**

Slave 設定で port 番号を指定すると、異なる機器からの Modbus アクセスを受けつけます。

設定: **SET_IP TCP Modbus　SRC 192 168 0 1**

EPSON 社製 RC コントローラ用の設定です。RC コントローラには Modbus-TCP に対応したメモリシェア機能があります。シェア領域は固定で以下のとおりです。対象は RC700,RC90,T3 等です。設定パラメーターとして SRC,SRC1～SRC3 があります。SRC と SRC1 はエリアが重なっていますので同時には使用できません。複数接続の場合は、SRC1～SRC3 を用いてください。

また、他の MPC を Modbus Slave に設定した状態で、Modbus SRC 接続を実施すると、MPC 間の Modbus メモリシェアとなります。この場合、Slave 側では、MBK(7000)～MBK(7099)が入力エリアとなり、MBK(7100)～MBK(7199)が出力エリアとなります。



設定: **SRC**

RC エリア	ビット	IN/OUT	WORD	MBK エリア
入力ポート	512～2111	64～263	32～131	MBK(7000)～MBK(7099)
出力ポート	512～2111	64～263	32～131	MBK(7100)～MBK(7199)

設定: **SRC1**

RC エリア	ビット	IN/OUT	WORD	MBK エリア
入力ポート	512～2543	64～317	32～158	MBK(7000)～MBK(7126)
出力ポート	512～2543	64～317	32～158	MBK(7130)～MBK(7256)

設定: **SRC2**

RC エリア	ビット	IN/OUT	WORD	MBK エリア
入力ポート	512～2543	64～317	32～158	MBK(7260)～MBK(7386)
出力ポート	512～2543	64～317	32～158	MBK(7388)～MBK(7514)

設定: **SRC3**

RC エリア	ビット	IN/OUT	WORD	MBK エリア
入力ポート	512～2543	64～317	32～158	MBK(7516)～MBK(7642)
出力ポート	512～2543	64～317	32～158	MBK(7644)～MBK(7770)

RC 側の設定は基本的なイーサネット設定の他に、フィールドバススレーブを有として、Modbus-TCP を選択します。詳細は、RC のマニュアルに従ってください。

設定: **SET_IP TCP Modbus AÐAM 192 168 0 1**

アドバンテック社 ADAM-6050,6052 を I/O として使用できます。

ADM エリア	ビット	IN/OUT	I/O
入力ポート	DI0～	301	200～207 (968~*
出力ポート	DO0~	303	8～15

*8bit 以上の入力は IN(121)で対応

MPC メモリシェア

MPC-3000 では UDP 通信を用いたメモリシェアを実装しています。また、データ保護には CRC16 を用いており、伝達されたデータは信頼性の高いものとなっています。

設定: スレーブ側 **SET_IP UÐP SHARE 3000**

設定: マスター側 **SET_IP UÐP SHARE 3000 REF_T¦100 192 168 0 13**

***REF_T** オプションは、マスター側で有効です。通信サイクルの間隔を msec で設定します。あまり短く設定すると、イーサネット通信が込み合い過ぎ全体としてうまく動作しません。50msec 以上が妥当です。

上記の設定では、マスター側の DT(7200)～DT(7723)のエリアがスレーブ側の同エリアに複写されます。複写サイクルは、100msec ごとになります。3 番目の引数、3000 はポート番号です。任意のポート番号を使用することができます。

マスター側でスレーブ側のデータを要求する場合は DT(7700)～DT(7723)に設定値を書き込みます。これらのデータセットは 0 が含まれると無効になります。

DT(7700) レスポンス要求 IP アドレス指定
DT(7701) 要求データエリア
DT(7702) 要求データ数
DT(7703) レスポンスディレイ(ms)

なお、UDP メモリシェア機能には、ブロードキャストを用いた 1: N シェア機能も用意されています。詳細は、別途技術情報を参照ください。

パケット通信

パケット通信は、とくにプロトコルを既定せず、文字列変数を用いてデータ交換をします。

TCP 通信	マスター	SET_IP TCP **PACKET** 9000 192 168 0 61	ポート番号と接続先アドレスを指定
	スレーブ	SET_IP TCP **PACKET** 8000	ポート番号のみ指定します
UDP 通信		SET_IP UDP **PACKET** 7000	ポート番号のみ指定します

マスター設定の場合は、指定アドレス、指定ポートに対して TCP コネクションを要求します。スレーブの場合は、TCP コネクションを待ちます。いずれの場合も、関数 **TCP(n)**によって判断します。n は、パワーオン時に与えられた TCP 番号です。

設定後パワーオンリセットすると MPC-3000 は以下のようなオープニングメッセージを出力します。

```
MKY =0
++++++++++++++++++
IPadrs = 192.168.0.14
 MASK  = 255.255.255.0
Gateway= 0.0.0.0
OFF Ðelay ACK
MACadrs= 70B3Ð553Ð00F
Ether Opened
TCP_STACK_RAM=11768
TCP Init Completed
Wake TCP
 IÐ:01 :  TASK 46 :  TCP  slave <TCP_X0$><TCP_R0$>
 IÐ:02 :  TASK 45 :  TCP master <TCP_X1$><TCP_R1$>
Wake UÐP
Wake Serial
USB_I/F  Ver1.12
iMPC-3000
#
```

また、オープニングメッセージ表示後、**SET_IP** コマンドを引数無で実行すると、現在の設定を表示します。この両方に現れる TCP_X0$,TCP_R0$文字列変数のラベルの数字が TCP 番号を表します。

```
#set_ip
IPadrs = 192.168.0.14
 MASK  = 255.255.255.0
Gateway= 0.0.0.0
OFF Ðelay ACK
MACadrs= 70B3Ð553Ð00F

1 : /  TCP   PACKET    8000 <TCP_X0$><TCP_R0$> Slave
2 : /  TCP   PACKET    9000 <TCP_X1$><TCP_R1$> Master 192.168.0.61 ref_time = 50
#
```

通信のやりとりは、TCP 番号ごとに指定された文字列と TCP(n)関数によっておこないます。

設定の有効化

RESET

SET_IP コマンドによる設定は、MPC-3000 への EEPROM への書き込みのみです。
設定を有効化するには、パワーオンリセットするか、**RESET** コマンドの実行が必要です。

SET_IP -1

書き込まれた設定値を消すには、設定番号を負の数として SET_IP を実行します。

SET_IP VOIÐ

設定値をすべて抹消するには、"SET_IP VOID"を実行します。
なお、設定を以下のようにプログラムに記述することができます。しかし、SET_IP は通常プログラム中では実行されません。これは設定値の有効化にはパワーオンリセットや **RESET** コマンドの実行が必要なためです。"SET_IP VOID"実行後 1 回のみプログラム中で実行されます。

```
10 SET_IP   192 168 0 14 255 255 255 0
20 SET_IP   TCP Telnet
30 SET_IP   UÐP MEWTOCOL7 8000
```

RUN IP_INIT

プログラム入れ替えで全てのパラメータを変更する場合は、**RUN** コマンドに引数 IP_INIT を与えます。

SET_IP VOID , RUN , RESET という一連の動作を実行します。
ただし、プログラムの実行をともなうために、装置が作動する条件はスタートボタン待ち等が必須です。こうした配慮が無いと不用意に軸やロボットが動作してしまいます。

シリアル通信の設定

タッチパネルの RS-232 通信は、TCP/IP とは関係がありませんが、同じバックグラウンド管理されるため、SET_IP コマンドで設定します。
ポート番号に続いてボーレートやキャラクタモードを指定できます。また、指定できる CH 番号は，1～14 と、MPC-3000 でサポートするすべての RS ポートに適用できます。
プロトコルは MEWNET のみです。

SET_IP Serial MEWNET 2	CH2 38400 B8NP で MEWNET 接続
SET_IP Serial MEWNET 8 115200 B7O	CH8　115200bps B7 Odd で接続

2-5 PACKET 通信関連コマンド

Packet 通信では以下のコマンドや関数を用います。

TCP(n)	TCP コネクション管理
TCP_Xn$,TCP_Rn$	パケット文字列 MAX500bytes
UDP_Xn$,UDP_Rn$	パケット文字列 MAX500bytes
INPUT_TCP n	INPUT#の TCP 版
IPC(Packet 文字列)	パケット文字列のサイズ
IPA(Packet 文字列)	送信元アドレス抽出
IP_CONV	IP アドレスのバイナリ化
SET_DEST	UDP 通信の宛先
PACKET	パケット通信汎用
PTR()	文字列のバイナリ読み出し
MODBUS,MODBUS()	Modbus-TCP パケット対応
Q3E	Q3E バイナリ・フレームの生成

*n は TCP 番号　0～6

TCP(n) : 関数

使用例:　IF TCP(0)==1 THEN
TCP コネクションの確認をします。n は TCP 番号です。TCP コネクションが成立していれば関数値として 1 を返します。切断していれば 0 を返します。なお、TCP コネクションの確立には一定の時間がかかります。

TCP(-1,n) : 関数

使用例: dmy=TCP(-1,n)
TCP 番号 n を TCP 切断する。dmy=TCP(-1,n) 。リターン値は常に 0 です。確認は、TCP(n)で行います。

TCP(-2,n) : 関数

使用例: dmy=TCP(-2,n)
TCP 番号 n を再接続する。dmy=TCP(-2,n) 。リターン値は常に 0 です。確認は、TCP(n)で行います。

TCP_Xn$,TCP_Rn$、UDP_Xn$,UDP_Rn$: 予約文字列

Packet 通信では設定順序と TCP/UDP の種別に従いこれらの文字列が使用されます。n はそれぞれ 0～6 までの数字文字です。サイズは一般文字列と異なり 512byte ありますが、送受信管理に 8byte 使用するためにユーザ用としては 500byte 程度としてください。

TCP 通信のもっとも単純な形式は、無手順 RS-232 通信を TCP 通信に置き換えたものです。この場合の通信は以下のように RS-232 通信と同様のプログラム構成となります。
使用例：

```
10         WAIT   TCP(0)==1
20         ĐO
30          WAIT   TCP_R0$!=""
40          PRINT   TCP_R0$
50        TCP_R0$=""
60        TCP_X0$="OK"
70         LOOP
#
```

注意として、受信後 TCP_R0$=""の実行が必要です。受信バッファがクリアされると次のメッセージを受け付けます。送信は、TCP_X0$="OK"のように文字列にデータを入れるのみです。
なお送信時の ESC シーケンスとして以下の機能を使用することができます。

¥r	CR コード
¥n	LF コード
¥t	TAB コード
¥0	ヌルコード
¥xhx	二桁のヘキサコードで表されるコード出力

INPUT_TCP 0 [EOL|chr] [TMOUT|m] 文字列 ：コマンド

実際の TCP 機器では、1 メッセージ、1 応答を守っていない場合もあり、前節の文字列処理だけでは対応できない場合があります。こうした場合は、INPUT_TCP を用います。

INPUT_TCP はパケット文字列(TCP_Rn$)より、指定したターミネータごとに文字列を切り出します。ターミネータ指定が無い場合は、CR(0x0d)をターミネータとして使用します。
対象文字列を読み取り終わると、rse_に 6 を返し、TCP_Rn$は初期化され、次のパケット待ち状態になります。以下の例は、１つのコマンド送信で４個のデータメッセージ(CR 区切り)が１つのパケットとして送信されてくる場合の例です。

使用例: 1

```
SET_IP 192 168 0 20 255 255 255 0 192 168 0 248
SET_IP TCP PACKET 9000 PC 192 168 0 58
ÐIM a(10)
ÐO
 WHILE LEN(TCP_R0$)<>0          /* 受信バッファクリア
 INPUT_TCP 0 CLR_BUF
 TIME 50
 WENÐ
 TCP_X0$="M" /* コマンド送信
 st$=""
 FILL a(0) 10
 FILL P(100) 10
 FOR dat=0 TO 4              /* データは<CR>区切りで 5 つある
 INPUT_TCP 0 EOL¦13 TMOUT¦10 a$
 IF rse_==1 THEN
 PRINT "TIME_OUT"
 P$(100)=""
 BREAK /* Exit FOR~NEXT
 ENÐ_IF
 P$(100+dat)=a$              /* この場合 P$は Max16 キャラ
 st$=st$+a$+" "                /* 後で GET_VAL で数値を取り出すため
 PR "P$(" 100+dat ")=" P$(100+dat)
 IF P$(100)<>"OK" THEN      /* NG とか ER など
 BREAK                         /* Exit FOR~NEXT
 ENÐ_IF
 NEXT
 IF P$(100)=="OK" THEN
 PR "st$=" st$
 GET_VAL st$ a(0)
 FOR i=0 TO dat-1
 PR "a(" i ")=" a(i)
 NEXT
 ENÐ_IF
 PR "---"
LOOP
```

次の例は、<CR><LF>で区切られた不特定数のメッセージが 1 つのパケットに収納された場合の対応例です。INPUT_TCP はタイムアウト 10 秒で動作し、その場合のみ DO～LOOP から抜けます。そうでない場合は、順次メーセージを<LF>区切りで取り出し、<CR>部を NULL に置き換えます。１つのパケットからデータを読み切ると、INPUT_TCP は自動的に受信バッファをクリアし次のパケットを受信するため、DO～LOOP の中でパケットの区切りとは無関係にメッセージを読み出すことができます。使用例 2 の"DUMP C_TCP_MSG$"は切り出されたメッセージの表示のみです。実際には、この行の代わりに必要な処理を記述します。

使用例 2:

```
SET_IP 192 168 0 20 255 255 255 0 192 168 0 248
SET_IP TCP PACKET 9000 PC 192 168 0 58

 INPUT_TCP   0 CLR_BUF :  rse_=0
 ÐO
rse_=0
  INPUT_TCP   0 EOL¦10 TMOUT¦10 C_TCP_MSG$
  IF   rse_==1 THEN
   PRINT   "timeout!" : BREAK
  ENÐ_IF
 e=LEN(C_TCP_MSG$)-1
  POKE   0 (C_TCP_MSG$+e)
  ÐUMP   C_TCP_MSG$
  PRINT   "rse_" rse_
 LOOP
```

IPC(Packet 文字列) : 関数

使用例: str_cnt=IPC(TCP_R0$)

パケットメッセージが ASCII 文字列で構成されている場合は、LEN()関数で文字列のサイズを知ることができますが、Modbus や MC_Protocol の通信では、内容がバイナリとなり 0 コードも内部に含まれるようになります。この場合にメッセージの長さを知るには、IPC()関数を用います。

IPA(Packet 文字列) : 関数

使用例: src_adr=IPA(UÐP_R0$)

TCP コネクションの場合は、相手アドレスが固定されていますが、UDP パケット通信の場合は、相手が不特定の場合があります。この場合は送られてきたパケットから送信元アドレスを知る必要がありますが、IPA()はこの IP アドレスを取得します。

IP_CONV : コマンド

使用例: IP_CONV 192 168 0 211 OppdIp

この例では変数 OppdIp に 192.168.0.211 がロング型整数として代入されます。

IP_CONV コマンドは通信には直接関係ありませんが以下のような変換を行います。

```
IP_CONV 192 168 0 195 plc   IP アドレスのバイナリ化 変数 plc に格納
IP_CONV IPA(UDP_R0$)        バイナリ IP アドレスの表示
```

SET_DEST adrs Packet 文字列 ：コマンド

UDP 通信の場合、MPC 側から先にパケットを送る必要があるとき、送出文字列に相手先アドレスを設定する必要があります。SET_DEST コマンドはこの場合に用います。
使用例は、次の PACKET コマンドの使用例を参照ください。

PACKET Packet 文字列 引数 1 引数 2 .... ：コマンド

PACKET コマンドは与えられた引数を順々にワードとして文字列に設定します。
コマンドが実行されると、自動的にメッセージが転送されます。
UDP 通信の場合は、コマンドの直前で、SET_DEST により送信先指定が必要な場合があります。
使用例(SET_DEST,PACKET):

```
IP_CONV 192 168 0 67  OppdIp
SET_DEST OppdIp UDP_X0$
PACKET UDP_X0$ &h4100 &h5701 &h0004 &h002A 1
WAIT IPC(UDP_R0$)!=0 /* 受信待ち
```

PTR(n,m) ：関数

```
使用例:  IF   PTR(2,0)!=&h0041 THEN
           TIME   100 : PRINT   "retry" : GOTO  *retry : END_IF
```

バイナリ通信の場合、Packet 送信後、Packet 文字列に収納されたデータは PTR で読み取ります。
読み取りには、バイト、ワード、ロングワードの区別がありますので、引数 n にそれぞれ 1,2,4 とバイト長を与えます。m はバイトで何番目かを指定します。また、エンディアンが異なる場合は、2,4 を-2,-4 とします。

MODBUS ：コマンド

Packet 通信で使用する MODBUS コマンドは、リモート I/O のように小規模なデータ設定を想定しています。このために、多くのデータを一度にセットするファンクション(16)には未対応です。多くのレジスタのデータ交換は、SET_IP で MODBUS 機能を指定して使用してください。
フォーマットは以下の二種類で、機能コード 3～6 に対応しています。

```
MODBUS TCP¦CHn Func    レジスタアドレス  書き込みデータ
MODBUS TCP¦CHn 4       レジスタアドレス  MBK 先頭番号  読み出し数<=80
```

```
使用例:  MODBUS   TCP¦0 3 &h0040 100 40
```

この例では、TCP|0 に対して、機能コード 3: Read Holding register を実行します。

読み出しレジスタの数 100 個とその値を格納する先頭、MBK(40)を指定しています
*読み出し数は、100 以下としてください。Modbus-TCP のパケット長は 256byte 以内のためです

使用例: MOÐBUS TCP¦0 4 &h0040 100 40
この例では、TCP|0 に対して、機能コード 4: Read input register を実行します。
読み出しレジスタの数 100 個とその値を格納する先頭、MBK(40)を指定しています。

使用例: MOÐBUS TCP¦0 5 &h0095 &hFF00 (&hFF00 → ON , &H0000 → OFF)
この例では TCP|0 に対して、機能コード 5 により ポート 150 番を ON します。
(Modbus では 0 を 1 番地とするため&H95 は 150 番とされます。

使用例: MOÐBUS TCP¦0 6 &h00C3 &h0011
この例では、TCP|0 に対して、機能コード 6: Preset single register を実行します。
アドレス&hC3、設定値&h11 を指定しています。

MODBUS(CHn;(Func,Adrs)) : 関数

読み出しレジスタ値が 1 つで良い場合は、この関数を使用します。
CHn が 0 の時は、CHn を省略し MODBUS(Func,Adrs)と記述できます。

MOÐBUS(4,0) 機能コード 4 で 0(1)番地を読み出します。
MOÐBUS(1;(4,1)) TCP1,機能コード 4 で 1(2)番地を読み出します。

Q3E : コマンド

三菱 PLC の MC プロトコルの 1 つである Q3E フレームを生成します(Header が 0x50 のタイプ)
アクセス方法はワードのみ、以下の 4 種類です。パケットが 512byte に限定されているので指定ワード数は 200 ワード程度としてください。また対応するエリアは、X,Y,M,B,D,W です。

WRITE_BULK 連続書込み
REAÐ_BULK 連続読み取り
WRITE_RNÐM ランダム書込み
REAÐ_RNÐM ランダム読み取り
WRITE_BIT ビット書込み

使用例: Q3E TCP¦n WRITE_BULK エリア アドレス ソース配列先頭 数
WRITE_BULK(連続書込み)対応例です。この例では CH0 に対して、"D"エリア 1000 番に aho(50)~aho(79)を書込みます 。

Q3E TCP¦0 WRITE_BULK "Ð" 1000 aho(50) 30

使用例: Q3E TCP¦n REAÐ_BULK エリア アドレス ソース配列先頭 数
REAÐ_BULK(連続読み込み)対応例です。この例では CH0 に対して、"D"エリア 1000 番を読

み取り aho(10)~aho(39)に書込み ます。

```
Q3E    TCP¦0 READ_BULK "D" 1000 aho(10) 30
```

*配列には、DIM 宣言した配列と、X(n)~Z(n),MBK が使用可能です。

使用例: Q3E TCP¦n WRITE_RNDM {エリア アドレス 値 }*n

WRITE_RNDM(ランダム書込み)対応例です。ここでは、D の 1000 番地に 100 W の 100 番地に 100 を書き込みます。

```
Q3E    TCP¦0 WRITE_RNDM "D" 1000 100 "W" 100 100
```

*引数の数は 14 個以内という制限があり、その範囲内で記述可能です。

使用例: Q3E TCP¦n READ_ RNDM {エリア アドレス}*n ソース配列先頭

READ_RNDM(ランダム読み込み)対応例です。ここでは、D の 1000,W の 100 を読み取り aho(10),aho(11)に代入します。

```
Q3E    TCP¦0 READ_RNDM "D" 1000 "W" 100 aho(10)
```

*引数の数は 14 個以内という制限があり、その範囲内の読み取りです。

使用例: Q3E TCP¦n WRITE_BIT エリア アドレス 数 {1 or 0 }*数

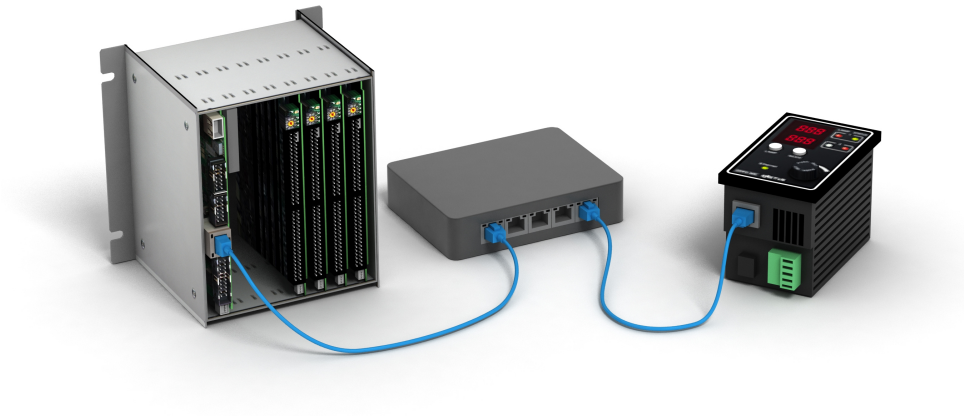
WRITE_BIT(ビット書込み)対応例です。以下では、M の 1000～1003 を ON/OFF/ON/OFF に設定します。

```
Q3E TCP¦0 WRITE_BIT "M" 1000 4  1 0 1 0
```

*引数の数は 14 個以内という制限があり、その範囲内の書き込みです。

使用例(オプテックス OPPD-30E,三菱 PLC,ADVANTEC ADAM)

以下に実際の機器に接続したサンプルプログラムを例示します。



UDP パケット通信 (オプテックス OPPD-30E)

```
SET_IP 192 168 0 20 255 255 255 0 192 168 0 248
SET_IP UDP PACKET 61440
```

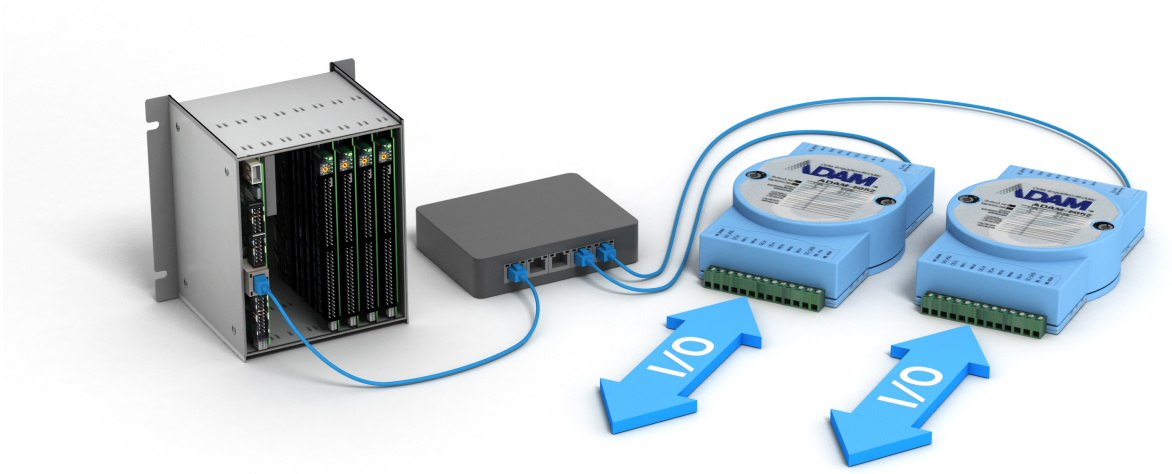
```
IP_CONV 192 168 0 211 OppdIp
IP_CONV 192 168 0 67 PcIp
SET_ÐEST OppdIp UÐP_X0$
PACKET UÐP_X0$ &h4100 &h5701 &h0004 &h002A 1
WAIT IPC(UÐP_R0$)!=0
C=0 : Ð=0
ÐO
*retry
UÐP_R0$=""
SET_ÐEST OppdIp UÐP_X0$
PACKET UÐP_X0$ &h4100 &h5700 &h0008 &h000C C &h001F Ð
timer_=10
WAIT IPC(UÐP_R0$)!=0 OR timer_==0
IF timer_==0 THEN
PRINT "No reply "
ENÐ
ENÐ_IF
IP_CONV IPA(UÐP_R0$)
ptr_=UÐP_R0$
IF PTR(2,0)!=&h0041 THEN
TIME 100
PRINT "retry"
GOTO *retry
ENÐ_IF
C=C+70
IF C>500 THEN
C=0
ENÐ_IF
Ð=Ð+170
IF Ð>500 THEN
Ð=0
ENÐ_IF
ÐO
UÐP_R0$=""
SET_ÐEST OppdIp UÐP_X0$
PACKET UÐP_X0$ &h4100 &h5200 &h0004 &h0039 &h0044
timer_=10
WAIT (IPC(UÐP_R0$)!=0)¦(timer_==0)
IF timer_==0 THEN
PR "TIME OUT OPPÐ"
ELSE
BREAK
ENÐ_IF
LOOP
ptr_=UÐP_R0$
PRINT "temp=" PTR(2,10) PTR(2,14)
ÐO
UÐP_R0$=""
```

```
SET_ĐEST PcIp UĐP_X0$
PACKET UĐP_X0$ &h0123 &h4567
Timer_=10
WAIT (IPC(UĐP_R0$)!=0)¦(timer_==0)
IF timer_==0 THEN
PR "TIME OUT PC"
ELSE
BREAK
ENĐ_IF
LOOP
IP_CONV IPA(UĐP_R0$)
ptr_=UĐP_R0$
a1=PTR(2,0)
a2=PTR(1,2)
a3=PTR(4,0)
PRINT "PC Đata=" HEX$(a1) HEX$(a2) HEX$(a3)
LOOP
```

Q3E 通信例　三菱シーケンサに接続

```
SET_IP 192 168 0 20 255 255 255 0 192 168 0 248
SET_IP TCP PACKET 9000 192 168 0 67
FILL P(10000) 4
FILL MBK(100) 3
FOR i=10000 TO 10003
 X(i)=i
NEXT
PRINT "WRITE_BULK"
Q3E TCP¦0 WRITE_BULK "Đ" 10000 X(10000) 4
PRINT "OK"
PRINT "REAĐ_BULK"
Q3E TCP¦0 REAĐ_BULK "Đ" 10000 Y(10000) 4
FOR i=10000 TO 10003
 PRINT "P(" i ")" P(i)
NEXT
PRINT "WRITE_RNĐM"
Q3E TCP¦0 WRITE_RNĐM "Đ" 10000 1234 "Đ" 10001 5678 "M" 100 8888
PRINT "OK"
PRINT "REAĐ_RNĐM"
Q3E TCP¦0 REAĐ_RNĐM "Đ" 10000 "Đ" 10001 "M" 100 MBK(100)
FOR i=100 TO 102
 PRINT "MBK(" i ")" MBK(i)
NEXT
```

Modbus I/O ユニットの制御(出力)例




```
SET_IP   TCP PACKET 502 192 168 0 196
TCP_R0$=""
TCP_X0$=""
pat=0
 ÐO
  TIME   100
  IF   aho(2)==0 THEN
   MOÐBUS   TCP¦0 6 &h012E 1¦pat¦(SW(0)*2)
   ELSE
   MOÐBUS   TCP¦0 6 &h012E pat
  ENÐ_IF
 LOOP
```

第３章　USB メモリと MicroMMC カード

3-1 USB メモリ

以下のコマンドや関数が使用できます。USB インターフェースには USB メモリとの相性があり、まれに対応できない場合があります。使用前に接続確認をしてください。確認は、USB(0)の実行と DIR コマンドによるファイル名参照をします。

USB(0)	USB ステータス関数	0 1	USB メモリが存在しない USB メモリ存在 USB()関数は、書込み、読み出し前に必ず実行してください。この確認がないと UWR,URD は動作しません
FILE$	予約文字変数	USB_READ,USB_WRITE の対象ファイル名	
USB_DEL	ファイル削除	USB_DEL　”ファイル名”　もしくは文字列変数	
USB_WRITE	１行書込み	FILE$で指定されたファイルに１行追加する 実行前に USB(0)で状態を判別	
USB_READ	１行読み込み	FILE$で指定されたファイルから１行読みだす 実行前に USB(0)で状態を判別	
USB_READ　-1	ポインタリセット	読み出しポイントをファイルの最初に戻します。	
DIR	ファイル名リスト		
TYPE	ファイル参照	TYPE “test.txt”	
USB_REN	ファイル名変更	USB_REN “123.TXT” “ABC.TXT”	
USB_PSAVE	点、MBK データ保存	アペンド保存です USB_PSAVE　P(1) 5000 "aa.p2k" USB_PSAVE　MBK(10) 1000 "aa.p2k"	
USB_PLOAD	点、MBK データ取り出し	上書きロードです。USB_PLOAD “aa.p2k”	
USB_LOAD	プログラムの読み出し	タッチパネルから制御可　*次節参照	

注) すべての USB コマンド、関数は１つのタスクで使用か、コマンドの衝突の無いように使用します。USB(0)関数で常時、USB の有無を検出し、別のタスクでデータの読み書きを行うような使い方はできません。 以下のように読み書きの直前で USB(0)関数を実行して読み書きの可否を判定します。

他のタスクでUSBの有無を知りたい場合は、以下の例のように変数に状態を複写します。

```
   IF   USB(0)==1  THEN
   USB_EXIST=1
A$="LOG_PASS "+ĐATE$(0)+" "+TIME$(0)+" "+STR$(CCC)+" "+aa$
  USB_WRITE   A$
ELSE
  USB_EXIST=0
ENĐ_IF
```

USB_LOAD

PCと連動して稼働させる装置では、MPCユーザプログラムの更新はPCを利用してのものとなりますが、タッチパネルとMPCだけの構成の場合、プログラムの更新をUSBメモリを介して行うことができます。

タッチパネルのDT7802(MBK(7802))に以下のような値を書き込むと対応するコマンドが実行されます。この機能により、タッチパネルの操作だけで、ユーザプログラムを更新することができます。

設定値(DT7802)	HEX表現	実行されるMPCコマンド	
-255	&Hffffff01	s_mbk -255 7802~Lng	CTRL_A
-254	&Hffffff02	s_mbk -254 7802~Lng	MPCINIT
-252	&Hffffff04	s_mbk -252 7802~Lng	ERASE
-248	&Hffffff08	s_mbk -248 7802~Lng	RUN
-240	&Hffffff10	s_mbk -240 7802~Lng	DIR 7650
-224	&Hffffff20	s_mbk -224 7802~Lng	USB_LOAD "AUTO_MC.F2K"
-192	&Hffffff40	s_mbk -192 7802~Lng	USB_LOAD "CHECK_MC.F2K"
-128	&Hffffff80	s_mbk -128 7802~Lng	USB_PLOAD "AUTO_MC.P2K"

*USBコマンド実行中は、設定値が&Hffffffxx --> &Hnnnn80xxとなり、コマンド終了で0となる
その他は、ただちに0になる
*DIR 7650では20個のファイル名がDT7650から6ワードごとに保存される
*ファイルのロード中は、DT7803がインクリメントし終了すると0となる
*ユーザプログラムは、一度MPCにロードして保存したものを使用してください。
*P2KファイルにはMBKデータもマージすることができます。

3-2 MMC カード

MPC-3000 には背面側にマイクロ SD カード用のソケットが付属します。このソケットにはマイクロ SD サイズの MMC カードを挿入して使用することができます。
MMC カードは、SPI モードを備えた SD カードで、2G までの容量に対応します。産業用として現在も製造市販されていますが、判別が難しい場合があります。
小社でオプションとして販売しておりますのでお問い合わせください。

コマンド	使用例
FILES	ファイル表示
MMC(0)	MMC マウントチェック
REMOVE	REMOVE “PP.P2K”
MMC_PSAVE	MMC_PSAVE P(1000) 300 "P_1.P2K" MMC_PSAVE MBK(500) 100 "P_2.P2K"
MMC_LOAD	MMC_LOAD FILE$ *プログラムロード
MMC_SAVE	MMC_SAVE FILE$ *プログラムセーブ
MMC_PLOAD	MMC_PLOAD "P_1.P2K"
MMC_WRITE	MMC_WRITE a$ *アペンドモードで書き込み
MMC_READ	MMC_READ a$ *EOF(1)で終端を検出
MMC_RENAME new$ old$	ファイル名の変更
CAT file$	ファイルのテキスト表示

注)MMC カードの読み書きは USB メモリに比べて高速です。用途としては、機種切替え用として多くの点データを保存し利用できることです。
すべての MMC コマンド、関数は 1 つのタスクで使用するか、コマンドの衝突の無いようにプログラムします。

Note:

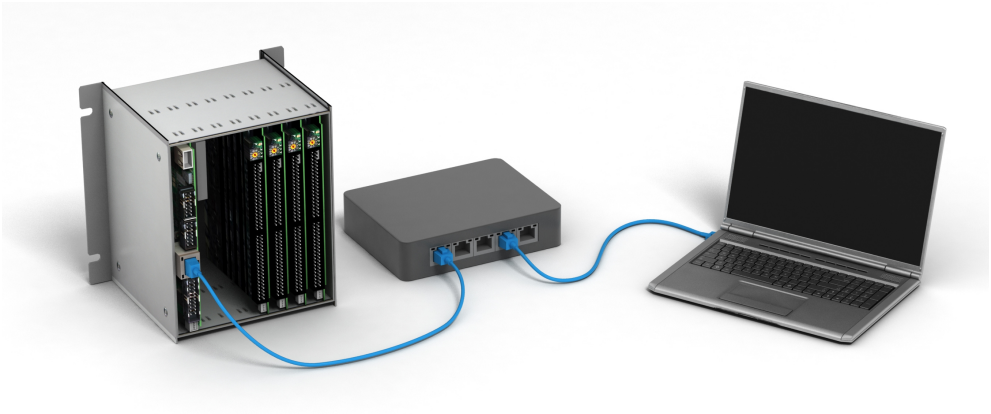
第４章　プログラミング

4-1 接続

　プログラミング接続は、J1 の RS-232 ポートを使用するか、イーサネット経由による Telnet 接続になります。ただし Telnet 接続は、RS-232 ポートでの SET_IP 設定が必須です。
　また、USB-RS を用いた RS-232 接続では、115200bps にも対応します。115200bps を使用する場合は、"SET_IP 115200"を実行しパワーオンリセットします。
38400bps に戻す場合は、"SET_IP 38400"を実行します。

RS 接続

Telnet 接続



実行

MPC でのプログミングは、コマンド行を書いて"RUN"コマンドを実行するだけです。コンパイルやダウンロードなどの手続きは不要です。また、MPC-3000 では、RUN は、プログラムをタスク 0 で実行するという意味ですので、RUN 実行後、プロンプトの応答があり、ただちに他のコマンドを実行することができます。従来のように応答タスクでインタプリタを実行したい場合は、引数に-1 を加えます。

```
RUN -1 *TEST
```

ツール

プログラミングツールには、以下の 3 種類があります。いずれも Windows 各バシージョン上で動作します。

FTMW2K	RS-232	スクリーンエディット
MPC_Monitor	RS-232	リアルタイムモニタ機能
MPC_Monitor_Telnet	Telnet	Lan 経由リアルタイムモニタ機能

初期段階のプログミングには FTMW2K でスクリーンエディトしながらのコーデングが向いています。MPC_Monitor は、稼働中にもさまざまな情報を参照できるため、全体動作が可能になってからの使用が効果的です。また、MPC_Monitor_Telnet は、MPC-3000 の Telnet ポートを介してのプログラミングツールのため、MPC-3000 にプログラミングケーブルを接続する必要がありません。稼働後のデバッグ、メンテナンスに効果的です。さらに、プログラム転送、保存が高速となります。

MPC_Monitor_Telnet では WAN を介しての Telnet 接続も可能です。これにより遠隔地からのモニター、保守に対応します。この場合、WAN 側のポートは 23 以外としてポート変換を行ってください。ポート 23 は、第三者が侵入を試みる最初のポートであるためです。

デバッグ

プログラム開発にはデバッグが必要です。プログラム規模が小さく単純である場合、FTMW2K の操作では、プログラム実行命令である RUN 後、以下のようなことができます。

MON	実行中の文番号表示
MON 1	実行中の文番号の連続表示(any key で停止)
CTRL+A	実行停止　停止行の表示
CTRL+M	実行停止　停止行の LIST 表示
MON 3	前回パワーオフ時の停止行　エラー履歴

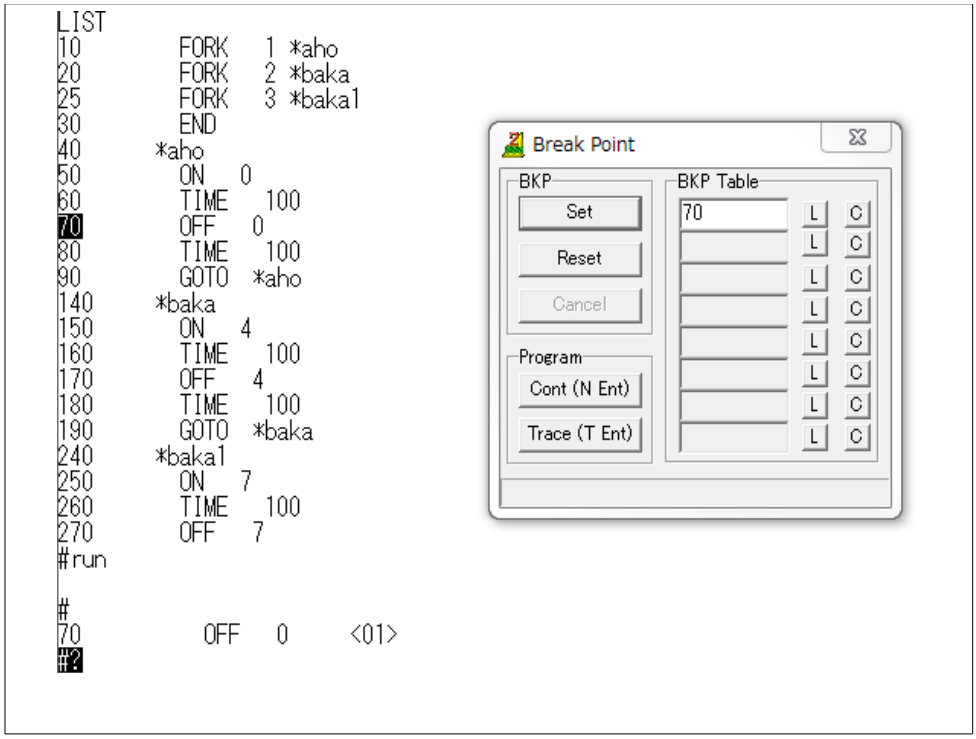
MON 3 は、稼働中に希に発生する停止に役立ちます。停止してパワーオンして再始動しても、停止時のログを保持します。このために、どのような原因で停止したのかを稼働中にも参照することができます。以下は、これらのデバッグコマンドを実際に実行した場合の FTMW2K での表示です。

```
#run
#mon
        *0_   [30]      *1    [60]      *2    [160]     *3    [260]
#
#mon 1
        *1    SLP [60]  *2    SLP [160] *3    SLP [260]
#
```

```
#CTRL+A
MKY =0
        *0_   [30]      *1    [60]      *2    [160]     *3    [260]
# CTRL+M
*0_      30         END
*1q      60         TIME   100
*2q      160        TIME   100
*3q      260        TIME   100
#mon 3
20190416 00153808
25->     タスクの二重起動です
 @Task 0 30      END
 @Task 1 70      OFF    0
 @Task 2 170     OFF    4
 @Task 3 170     OFF    4
#
```

ブレークポイント

　実行プログラムにはブレークポイントを設定することができます。
ブレークポイントで停止すると、その場所で変数の参照をしたり、ワンステップ実行などをすることができます。以下は、FTMW2K でのブレークポイント使用画面です。

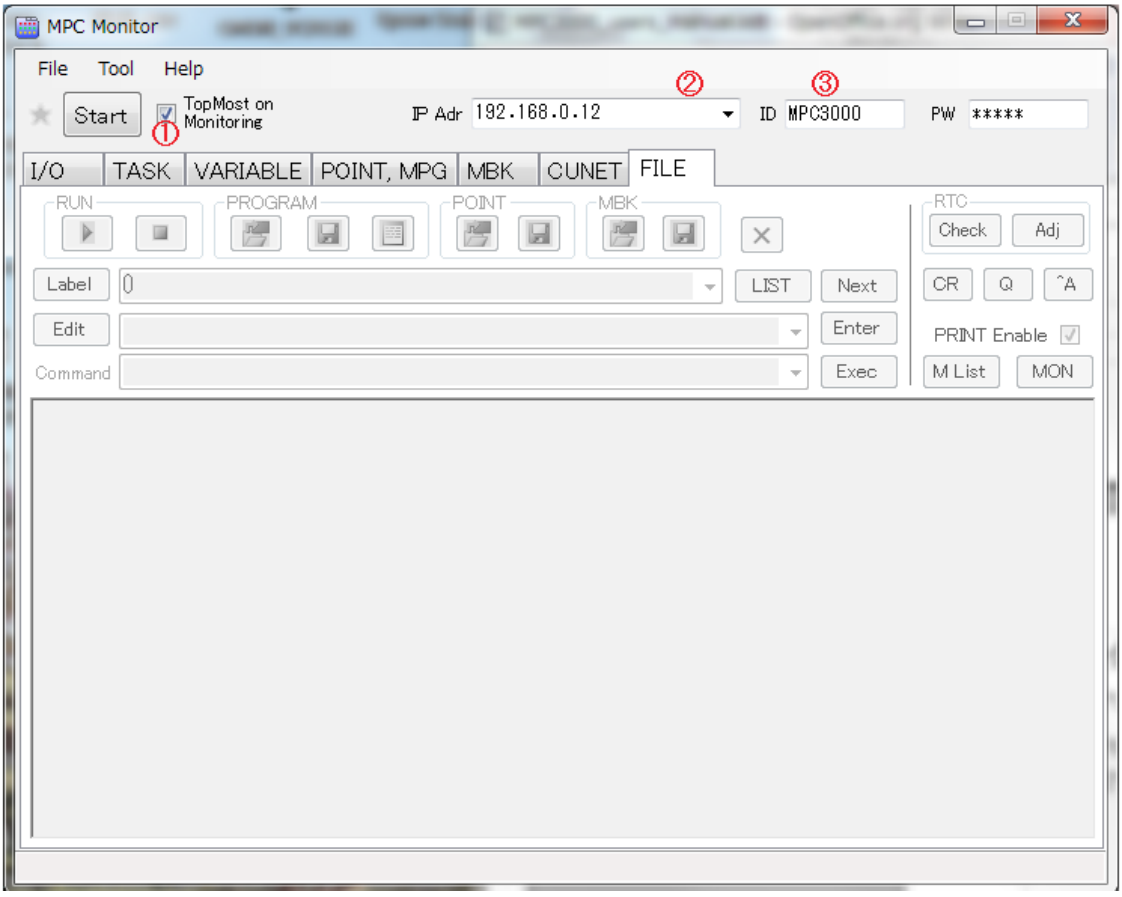


4-2 MPC_Monitor_Telnet

MPC_Monitor_Telnet は、イーサネット経由で MPC のプログラムのセーブロード、実行状態の確認を行うことができます。このため、装置に組み込まれ試用から稼働状態でのプログラム実行状態の監視、プログラムの修正、保存、更新を行うことができます。

また、工場内ルータを適切に設定することにより、遠隔地から同様の作業を行うことも可能です。ただ、この場合、Telnet の項で述べたように、Telnet の標準ポートであるポート 23 は、WAN に対して開かないでください。ポート変換を実施することにより、ハッキングからシステムを守ることができます。

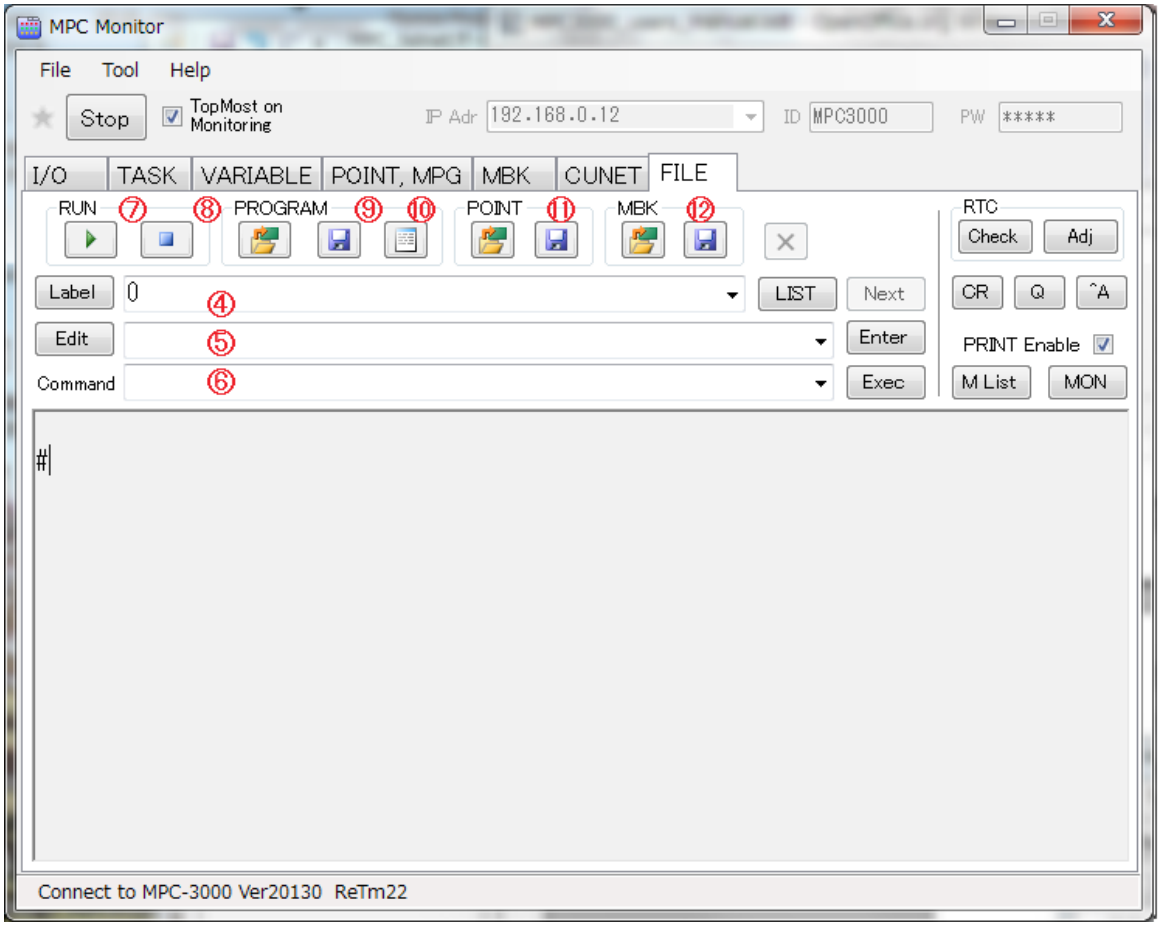
オープニング画面



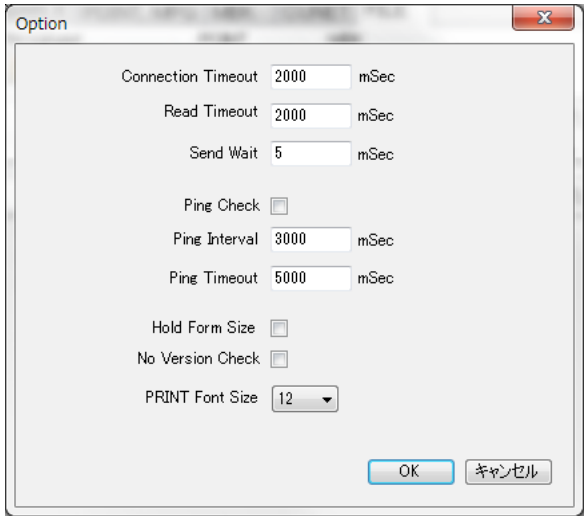
MPC_Monitor_Telnet は、IP アドレスで MPC を指定します。以下の例では、192.168.0.12 の IP アドレスの MPC に対して Telnet 接続を実行します。IP アドレスは②の枠に入力します。WAN からの接続で、ポート番号を変更している場合は、153.142.251.41:8465 のように IP アドレスの後ろに、:（コロン）を与えてポート番号を記述します。③はログイン名とパスワードの入力 BOX です。

呼びかけられる MPC には、” SET_IP　TCP Telnet “が設定されていなければなりません。

準備ができたら①のスタートボタンを押します。接続が成功すると、MPC の#プロンプトが表示されます。



正しく設定したのに接続できない場合は、メニューの Help->Option を開いて以下の画面を確認します。Connection Timeout,Read Timeout はいずれも 3000msec が推奨です。Send Wait は 5msec ～10msec とします。



接続が完了したら実際の操作です。

④　プログラムのラベルや行番を指定します。指定した上で LIST ボタンを押します。

⑤　行単位でプログラムの修正をおこなうことができます。挿入の場合は、行番号とコマンドを入力して Enter を押します。すでにある行を変更する場合は、表示行を逆クリックして表示されたメニューの"キャレット行の編集"をクリックすると EditBOX に複写されます。

⑥　コマンドをここに設定して Exec ボタンを押すと、実行されます。

⑦　プログラムを実行します。

⑧　プログラムを全停止させます。

⑨　プログラムをロード、あるいは保存します。

⑩　プログラムを編集します。

⑪　点データの読込・保存

⑫　MBK エリア(DT エリア)のデータ読込・保存

Check　MPC の RTC の時間を表示ます。

Adj　PC の時間を MPC の RTC に設定します。

CR　CR のみ出力します。s_mbk など連続表示に対応

Q　'Q'のみ出力します。s_mbk など連続表示に対応

^A　CTRL_A を与えて、ソフトリセットを発生させます。

MLIST　実行中、あるいは停止後の行を表示します。

MON　実行中、あるいは停止後の行番号のみを表示します。

*⑨ この機能は、MPC_Telnet 1.03,ファームウェア BL/I 2.01_30 以後で高速化されています。
　PC へのプログラムセーブは Windows8,10 では、高速になりますが、Windows7 では効果がありません。高速化するには、使用中の Windows7 PC のレジストリの修正が必要となります。以下に示すレジストリキィーに"TcpAckFrequency"=dword:00000001 を加筆します。

HKEY_LOCAL_MACHINE\SYSTEM\ControlSet0XX\services\Tcpip\Parameters\Interfaces\{B04CE29C-A42E-4E57-BEB1-C74026811AA2}]
PC がどのコントロールセットを使用しているかは、以下の Current の値で決められています。

```
[HKEY_LOCAL_MACHINE\SYSTEM\Select]
"Current"=dword:00000001
"Ðefault"=dword:00000001
"Failed"=dword:00000000
"LastKnownGood"=dword:00000002
```

この場合は ControlSet001 が選択されています。

プログラムロードは、MPC のフラッシュメモリにファイルのバルク転送をすることによって高速化されています。

オプションの MMC カードを装着し、MMC_LOAD$の文字列を"FLASH.F2K"以外に設定するとプログラムロードは MMC 経由となります。この場合、転送されたプログラムは MMC_LOAD$で指定されたファイル名に一旦転送されます。速度は、少し遅くなりますが、転送したプログラムの控えをＭＭＣカードに残すことができます。

MMC_LOAD$は MPCINIT で"FLASH.F2K"に初期化されています。

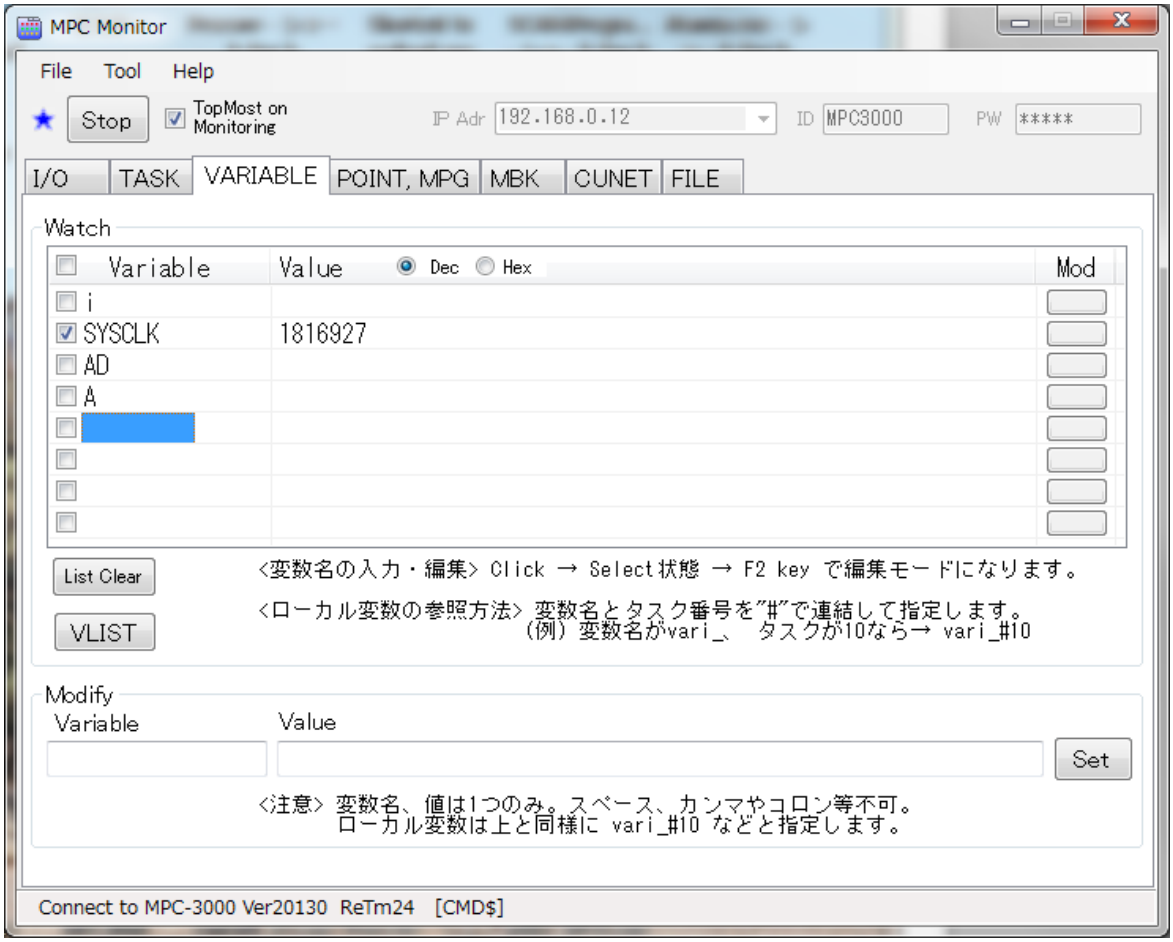
変数参照

MPC_Monitor_Telnet では、⑩のプログラムエディタを用いて、プログラムを作成し、⑨のプログラムロードでプログラムを転送し、⑦で実行するという手順になります。

実行状況は、I/O,TASK,VARIABLE,POINT,MPG,MBK.CUNET の各タグをクリックすることにより参照できます。ただし、VALIABLE タグは変数モニターで以下のような設定が必要です。

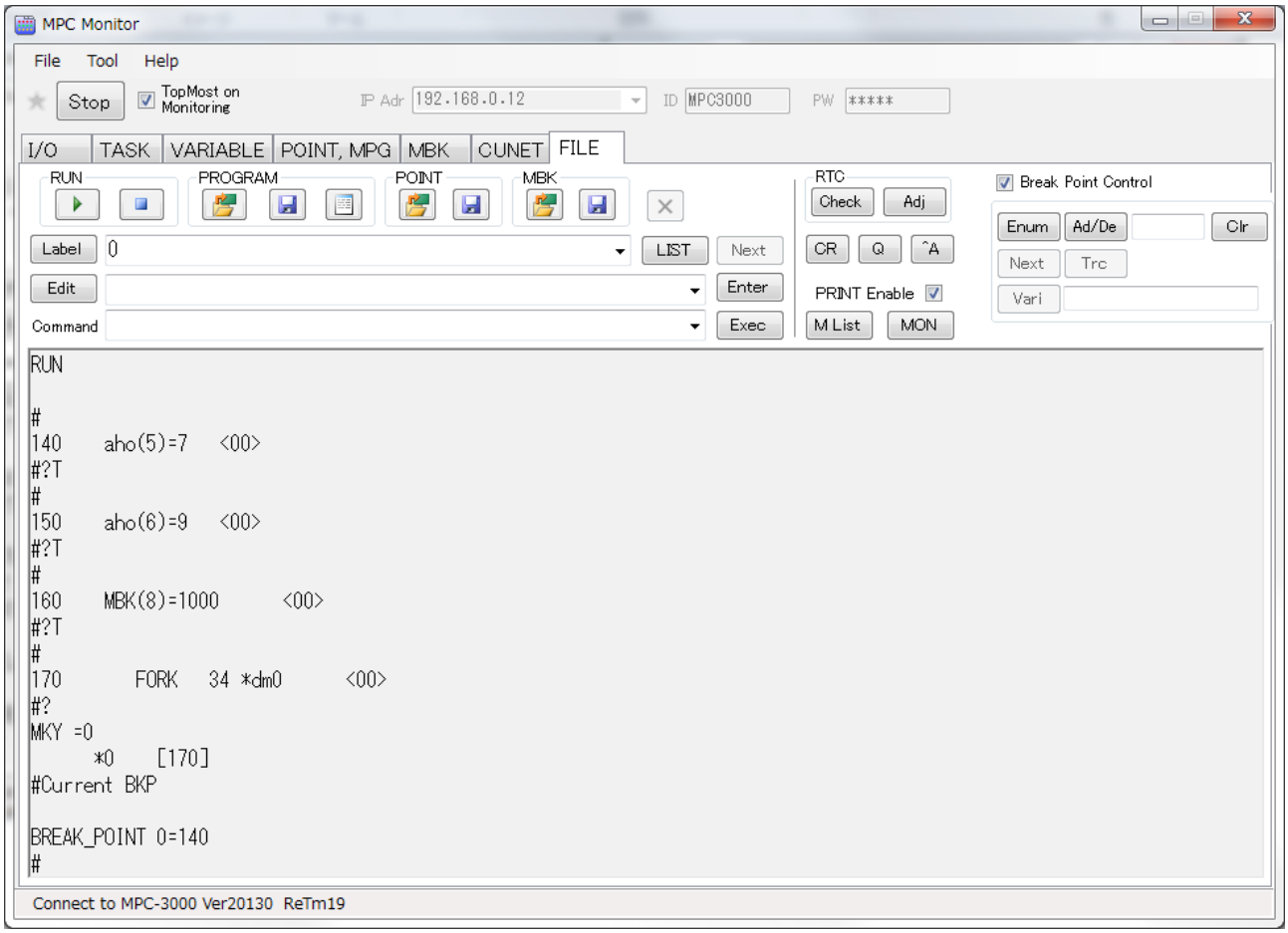
Variable の項でマウスを右クリックすると、変数ラベルが入力可能となります。そこにラベルを入力して、Watch のチェックをいれると、リアルタイムで変数値を参照することができます。

以下例では SYSCLK を設定していますが、1msec 毎にインクリメントする様子を参照できます。



デバッグ

　ブレークポイントを利用したデバッグ機能も備えられています。
デバッグは FILE タグでのみ使用可能です。右枠をマウスでドラッグすると、左のようにデバッグツールが現れます。Break Point Control にチェックをいれると、ブレークポイントの設定、参照を行うことができます。
　プログラムを表示して、カーソルを当て Ad/De を押すとブレークポイントが設定されます。
Next で再実行 Trc でワンステップ実行となります。
現在のブレークポイントの表示は、Enum です。Clr でブレークポイントは解除されます。
　MPC_Monitor_Telnet の詳しい説明は、当社 Web サイトで用意しております。
詳説は、そちらを参照ください。

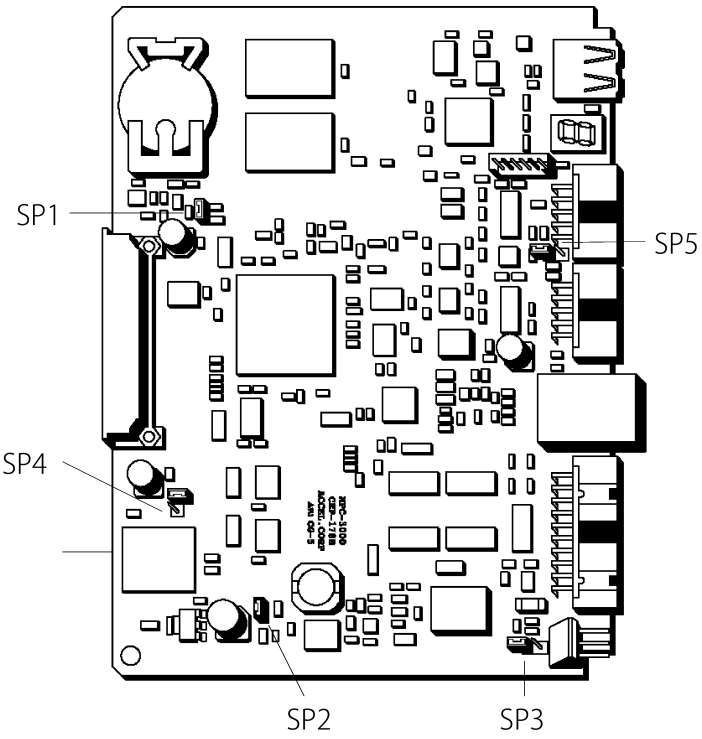


第 5 章　MPC-3000 ハードウェア

5-1 仕様

CPU	R5F72167ADFA SH2A 内臓　ROM 512K　 RAM 64K
プログラムエリア	1Mbyte
点データ	32000 ポイント
RS-232C	CH1,CH2,CH18 ユーザ用　4800～115200bps
TASK 数	48 タスク、ただし 47 はシステム使用。SET_IP でタスク 46 より降順使用
RTC	RX-8035LC　カレンダ時計機能
I/O	入力 8 点、出力 8 点　入力は 2 線式対応
7SEG	1 桁　PR_LCD によって表示可
USB	USB メモリ対応　PIC24FJ256GB106-I/PT 使用
MicroSD	micro MMC カード使用可
イーサネット	RJ-45 10M/100M 対応
内部電源	DC5V 2.0A まで供給可　16slot 可　(自己消費 300mA 以下)
バッテリ	CR-2032
電源	DC24V 単一

5-2 ハード構成



ンターフェース

ングポート

-232　2CH

ト(RJ-45)

出力8点(NPN)

r

mA)

・SP1　オープンで sysld2k 有効
・SP2　MPC-3000 内部電源供給
・SP3　FG 結合
・SP4　DC5V 拡張供給
・SP5　RS-485 ターミネータ　ショートで有効

① J1　RS-232

1	SG	2	TXD
3	RXD	4	SG
5	MAN	6	P5
7	SG	8	TX1
9	RX1	10	FG

② J6

1	FG	2	TX2
3	RX2	4	P5
5	SDB*	6	SDA*
7	SG	8	TX18
9	RX18	10	NC

*SDA，SDB は RS485 です。RS485 は CH2 と扱われ RS485 使用時は CH2　RS232 は使用できません。
*RS232,485 は、J3 電源とアイソレーションされています。

③ J4

1	SW(192)	2	SW(193)
3	SW(194)	4	SW(195)
5	SW(196)	6	SW(197)
7	SW(198)	8	SW(199)
9	ON0	10	ON1
11	ON2	12	ON3
13	ON4	14	ON5
15	ON6	16	ON7
17	+DC24	18	+DC24
19	GND	20	GND

④ J3

1	+DC24
2	GND
3	FG

*DC24V はステップダウンレギュレータにより、USB を含む内部回路に給電されます。USB の SG,VCC は他回路と接続できません。

5-3 温度管理

MPC-3000 の動作温度は、0℃から 50℃の範囲です。
結露の恐れのある環境下では使用しないでください。
また、収納 BOX の内部温度が動作温度範囲を満たしている場合でも、MPC-3000 付近の空気の流れが停止していると、自己発熱により動作温度を越えてしまいます。
MPC-3000 収納 BOX 内の空冷循環には十分注意してください。
　RACK-V16S では側板に CPU 空冷用として、オリエンタル MDS410-***が取り付け可能となっています。(***は電圧選択など)

MPC-3000 ユーザーズマニュアル
2019 年 6 月　　第 1 版
発行責任者　　横田　隆一
発行所　　株式会社アクセル
〒391-0005
長野県茅野市仲町 16-32 トウブビル 5F
TEL:0266-72-8465　FAX:0266-72-8436
E-mail　sales-ac@accelmpc.co.jp
http://www.accelmpc.co.jp
企画・編集　　フリーシステム