MPC-3000,3200 イーサネット機器接続例

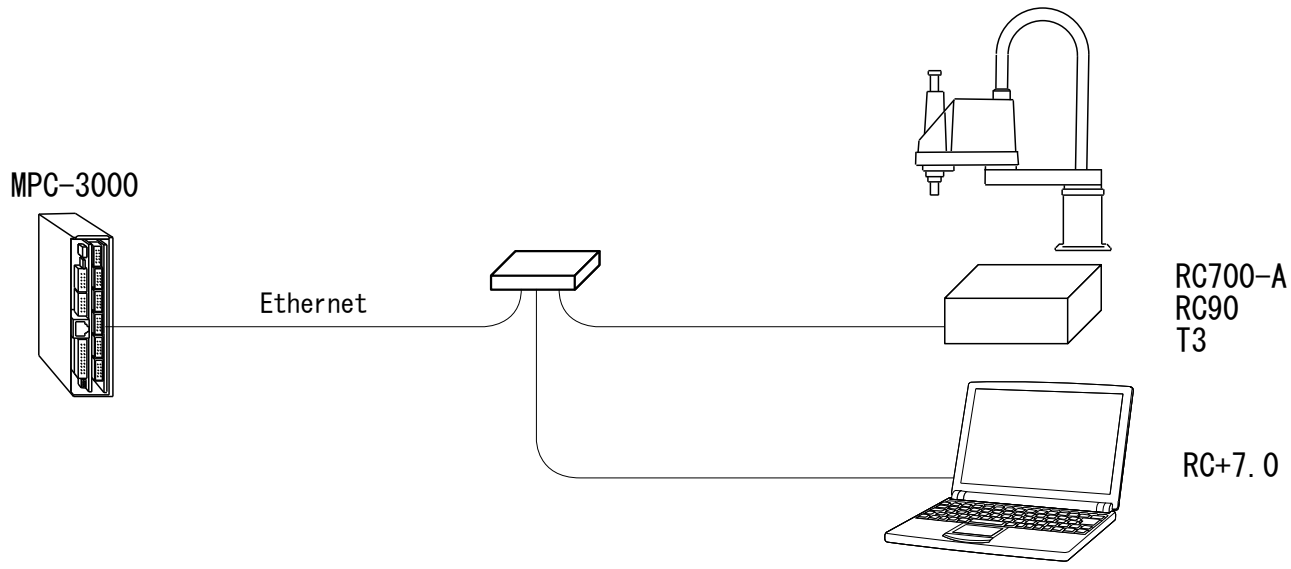
(イーサネット以外の新機能も含みます)

ACCEL ti2k-180828 (LastUpdate Jan 23, 2024)

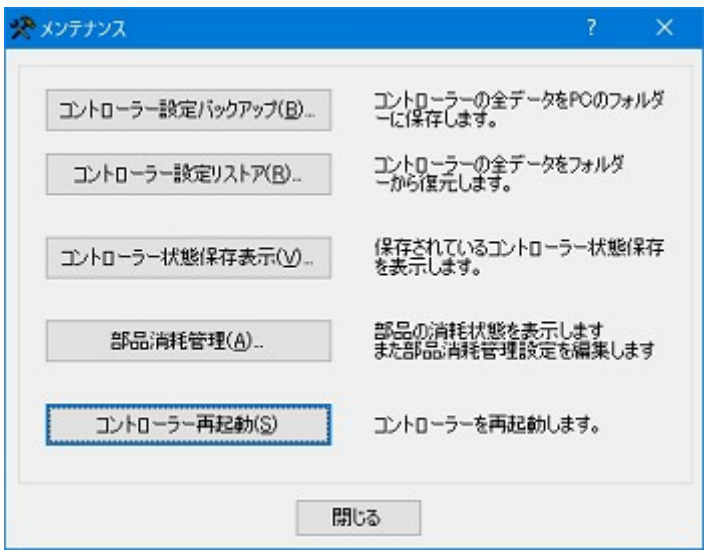
EPSON RC700 Modbus接続の設定
MPC同士のメモリシェア
MCプロトコル通信
三菱シーケンサQ03UDECPUとの通信
三菱ロボットコントローラ CR800-Dとの通信
Proface GP4301 MEWTOCOL7接続の設定例
KEYENCE VT3 MEWTOCOL7接続の設定例
MPC Monitor Telnetの設定
OMRON 画像処理システム FHシリーズ
MPCとパソコンの直結
SET_IP 設定の更新方法
LED照明コントローラ OPPD とのUDP通信
「¥」「"」の送信、受信(キャラクタ置換)
RS-232 MEWNET 接続
Advantech ADAM-6052 Modbus/TCP 接続
連続パケット受信と文字列分割例
Modbus/TCPによるMPC～パソコン間メモリシェア
MPCにアクセスするModbus/TCPプロトコル例
MMC(Multi Media Card)アクセス
タッチパネルコマンド
ファームウェアのUpdate手順
アプリケーション実例

※この資料はA3サイズで作成しています。

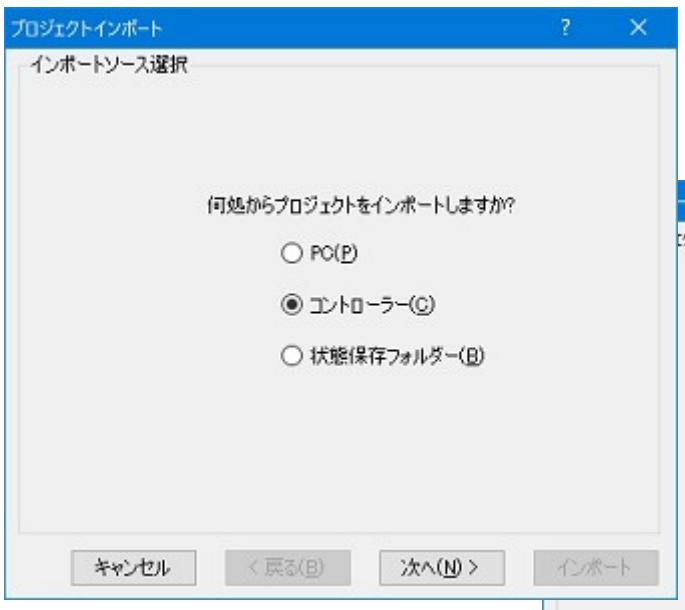
EPSON RC700 Modbus接続の設定



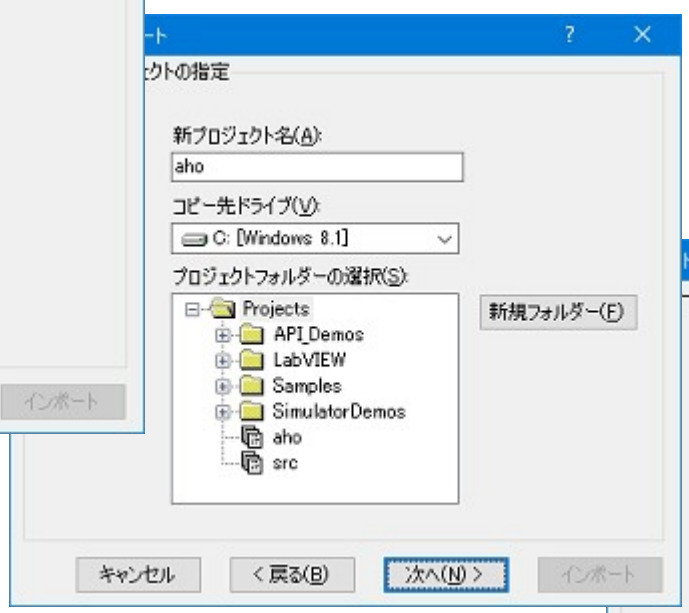
■ RC の設定

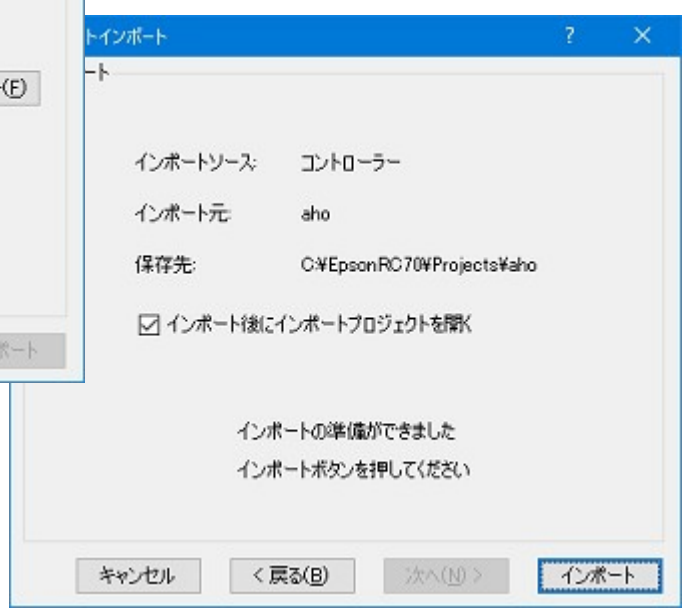


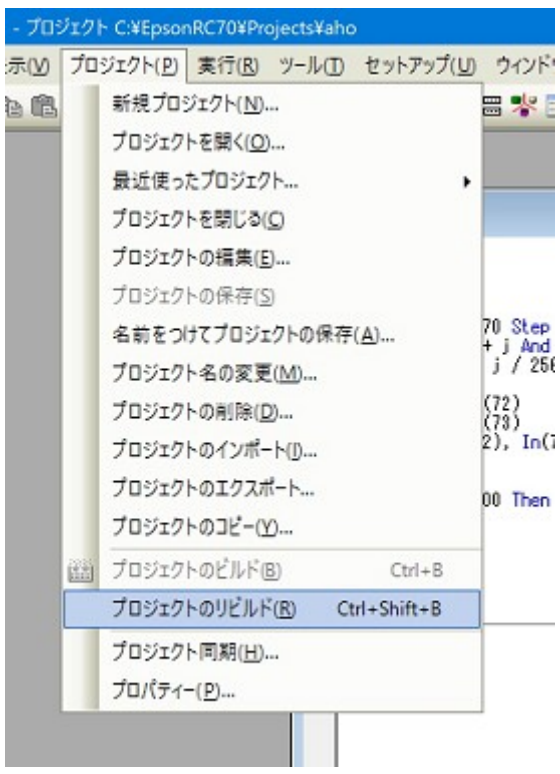
MPCをパワーオフ等して通信が途絶えたときは、
RCを再起動しなければならない。



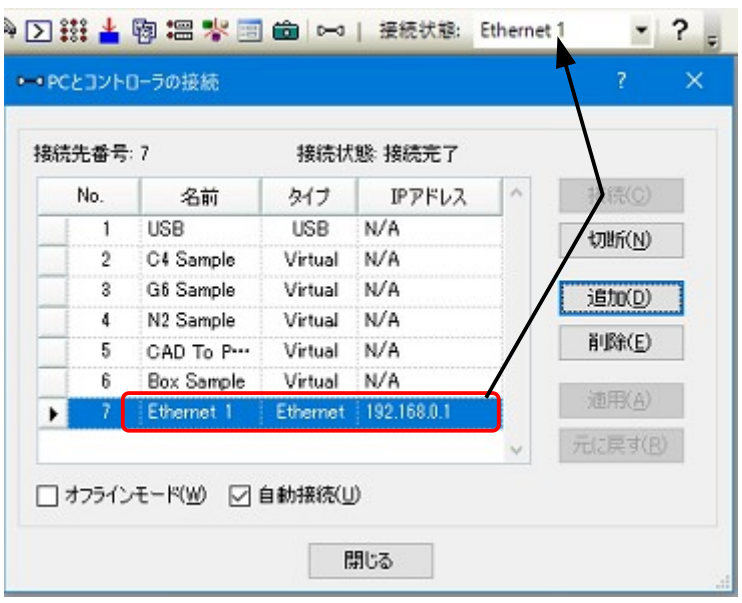
違うPCの RC+7.0 でプロジェクトを実行する場合、インポートとリビルドをする。





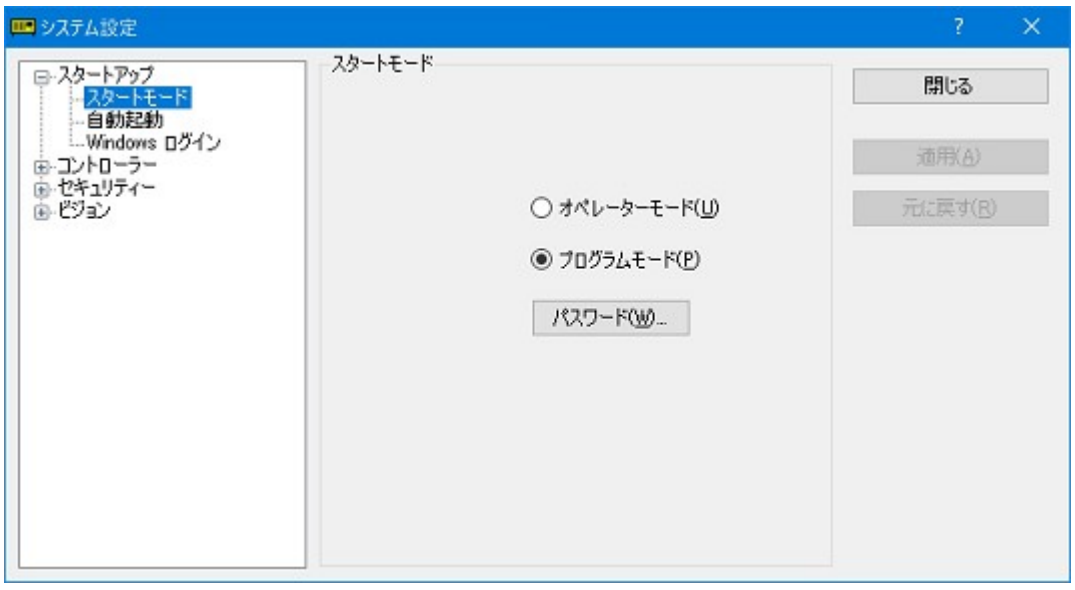


セットアップ ＞ PCとコントローラーの接続

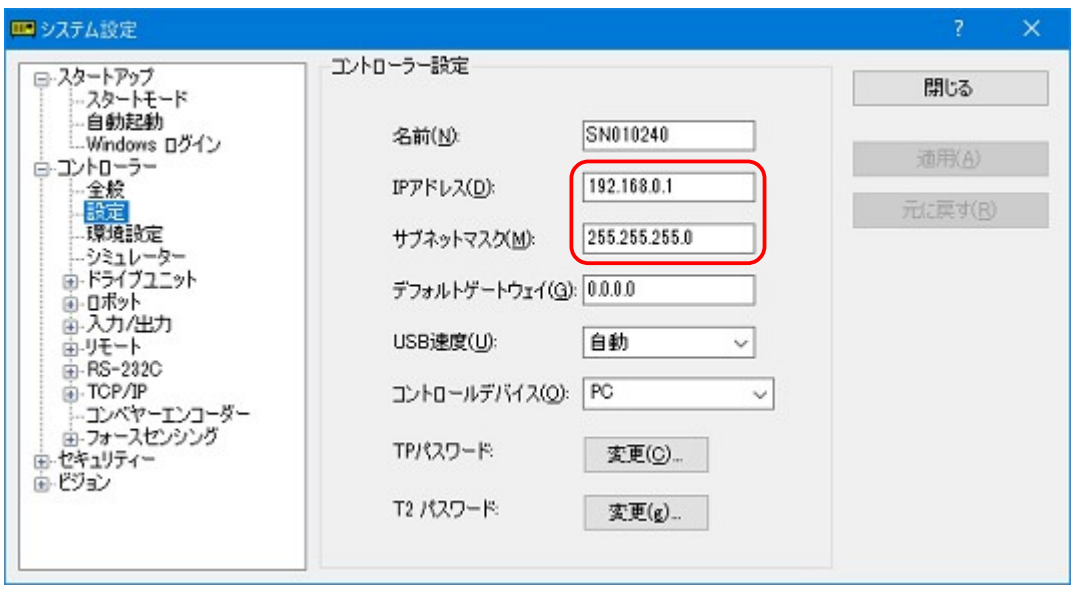


Ethernet → 接続

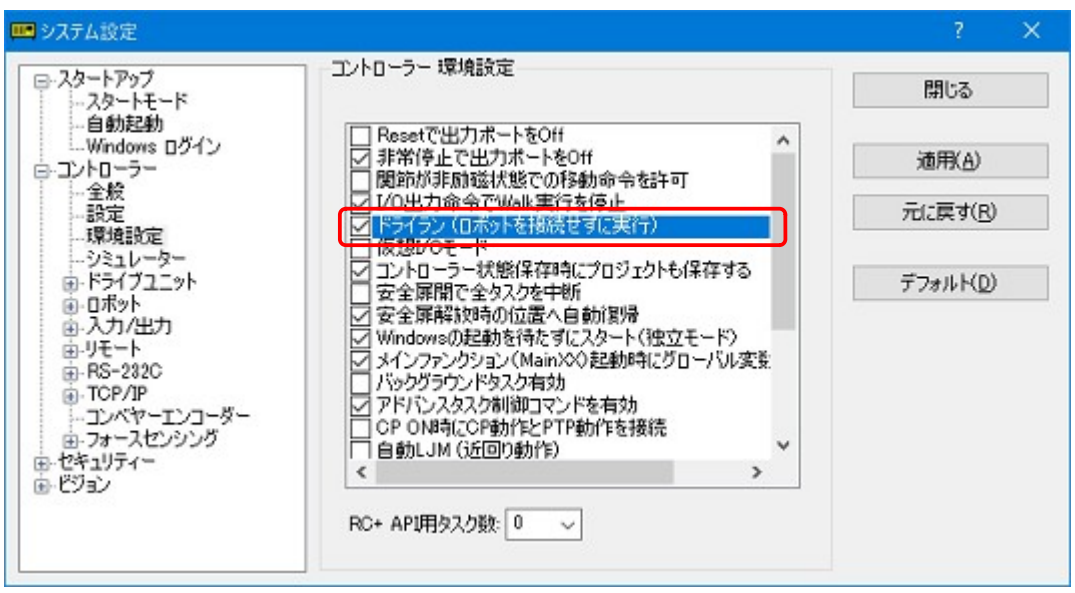
セットアップ ＞ システム設定＞スタートアップ



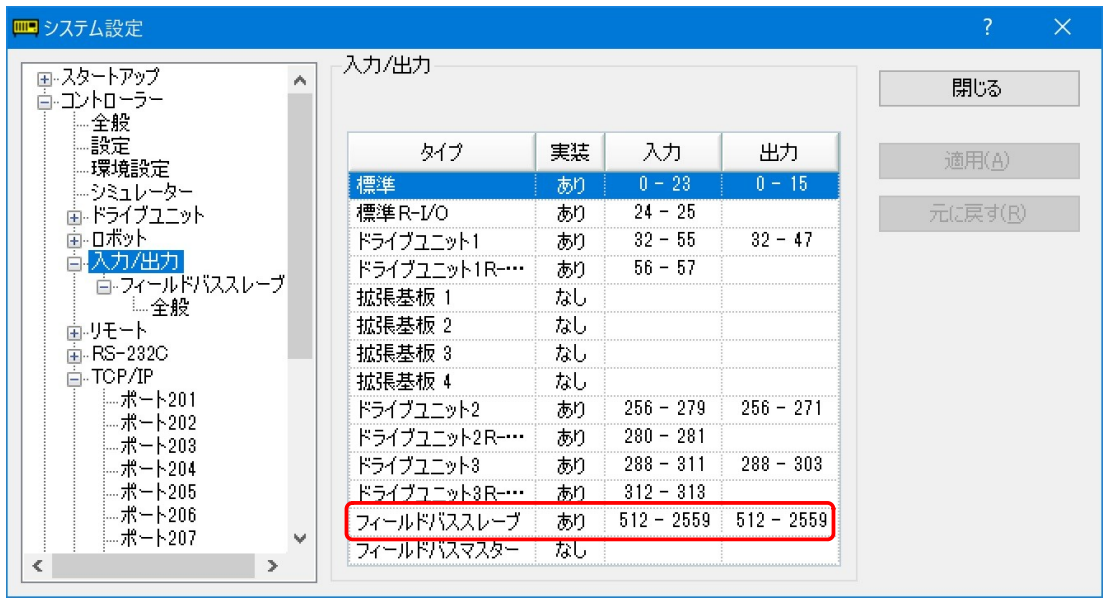
セットアップ ＞ システム設定＞コントローラー＞設定

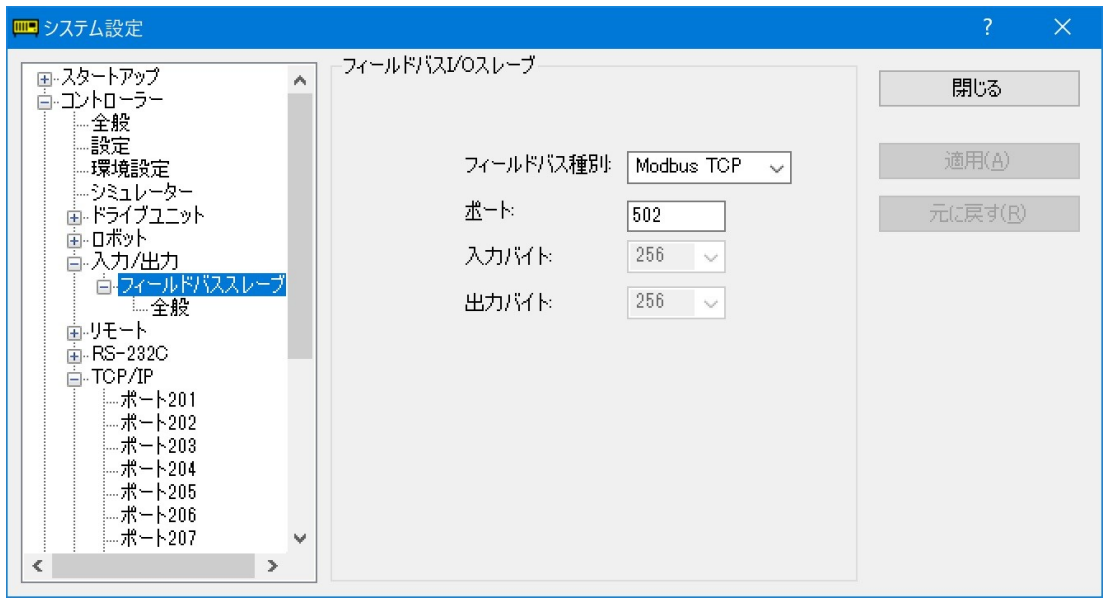


セットアップ ＞ システム設定＞コントローラー＞環境設定

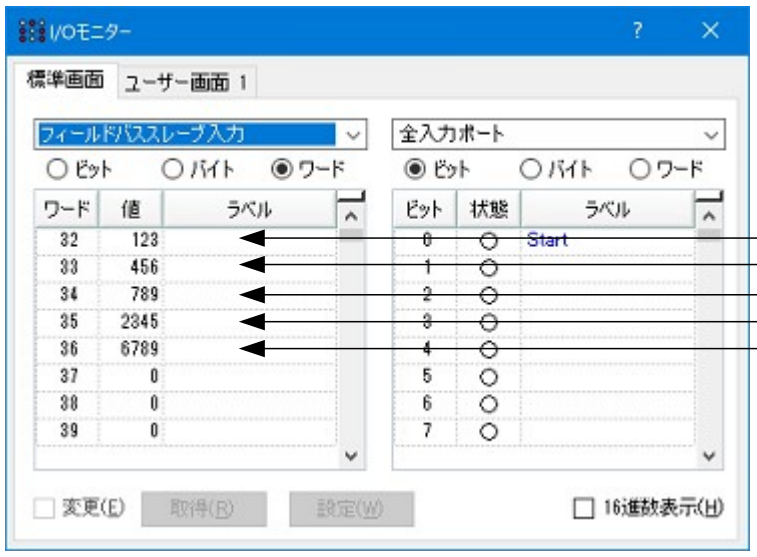


ロボット本体無しで実行するとき


システム設定
スタートアップ
コントローラー
全般
設定
環境設定
シミュレーター
ドライブユニット
ロボット
入力/出力
フィールドバススレーブ
全般
リモート
RS-232C
TCP/IP
ポート201
ポート202
ポート203
ポート204
ポート205
ポート206
ポート207
入力/出力
閉じる
適用(A)
元に戻す(R)
タイプ
実装
入力
出力
標準
あり
0 - 23
0 - 15
標準 R-I/O
あり
24 - 25
ドライブユニット1
あり
32 - 55
32 - 47
ドライブユニット1R-…
あり
56 - 57
拡張基板 1
なし
拡張基板 2
なし
拡張基板 3
なし
拡張基板 4
なし
ドライブユニット2
あり
256 - 279
256 - 271
ドライブユニット2R-…
あり
280 - 281
ドライブユニット3
あり
288 - 311
288 - 303
ドライブユニット3R-…
あり
312 - 313
フィールドバススレーブ
あり
512 - 2559
512 - 2559
フィールドバスマスター
なし


システム設定
スタートアップ
コントローラー
全般
設定
環境設定
シミュレーター
ドライブユニット
ロボット
入力/出力
フィールドバススレーブ
全般
リモート
RS-232C
TCP/IP
ポート201
ポート202
ポート203
ポート204
ポート205
ポート206
ポート207
フィールドバスI/Oスレーブ
フィールドバス種別:
Modbus TCP
ポート:
502
入力バイト:
256
出力バイト:
256
閉じる
適用(A)
元に戻す(R)

■ メモリ共有イメージ



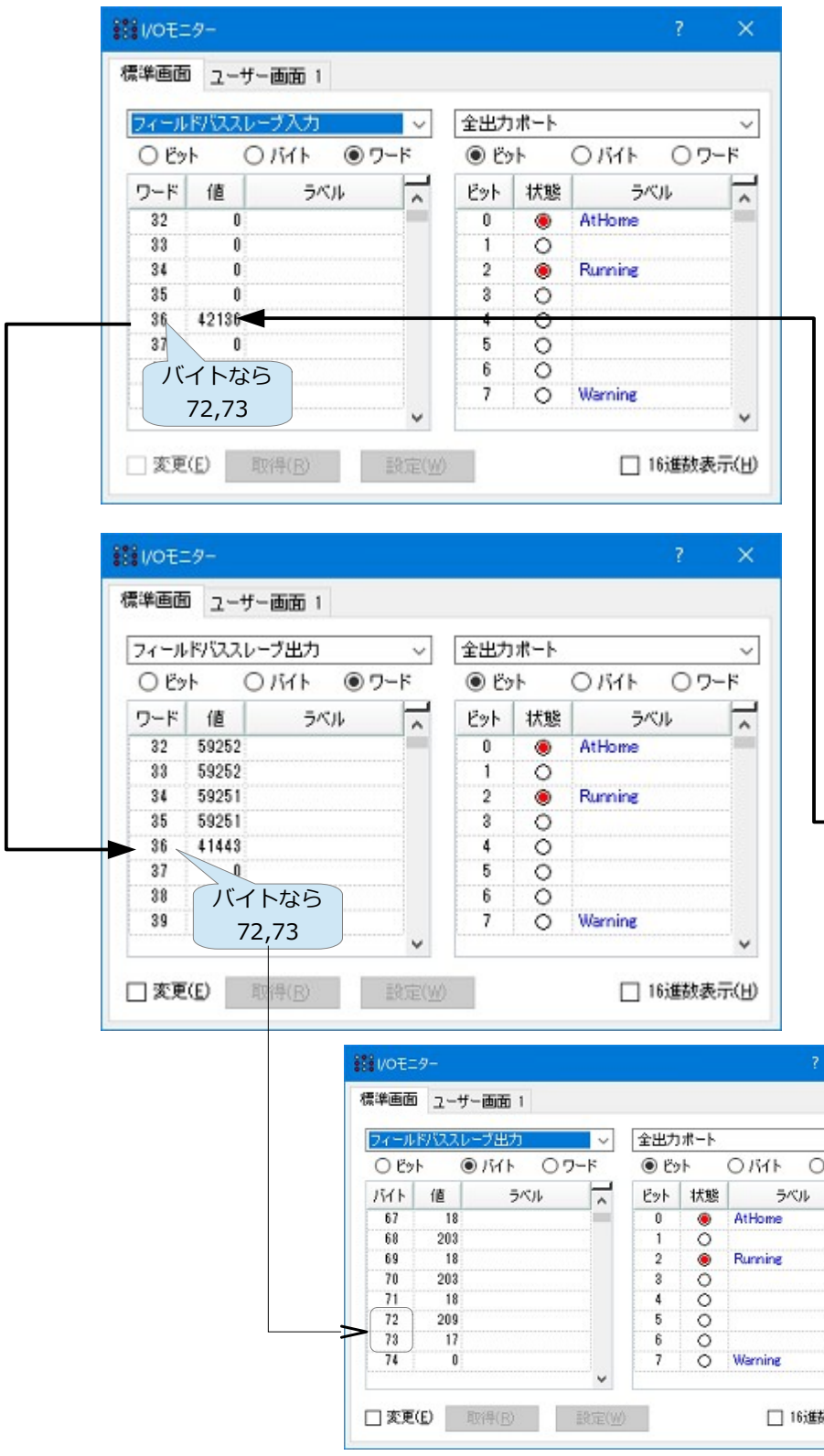
MPCのMBKエリア 7000~7099 が
RCのフィールドバススレーブ入力に転送される

MBK(n)	Value D H
7000	123
7001	456
7002	789
7003	2345
7004	6789
7005	0
7006	0
7007	0

RCのフィールドバススレーブ出力が
MPCの MBKエリア 7100~7199 に転送される

MBK(n)	Value D H
7100	321
7101	654
7102	987
7103	5432
7104	9876
7105	0
7106	0
7107	0

■ プログラム実行例



RCプログラム

Main.prg

```
Function main
Int32 i, j
j = 0
Do
For i = 64 To 70 Step 2
    Out i, 10 + j And 255
    Out i + 1, j / 256
    Wait 0.01
    Out 72, In(72)
    Out 73, In(73)
    Print In(72), In(73)
Next
    j = j + 1
    If j > 60000 Then
        j = 0
    EndIf

Loop
```

MPCプログラム

```
*BASE0
 mm=0
  DO
   IF    jo&4==0 THEN  : ON   70001 : ELSE   : OFF   70001 : END_IF
   INC    jo
   FOR    kk=0 TO 7
    ON    kk : WAIT   SW(kk+192)
    TIME    10
    OFF    kk : WAIT   @SW(kk+192)
 MBK(7004)=mm
    WAIT    MBK(7104)==mm :   mm=mm+1
    IF    mm>60000 THEN   :   mm=0 : END_IF
   NEXT
  LOOP
```

※プログラム実行中のスクリーンキャプチャなので、各数値は一致しません。

■ MPC-3000 プログラムと接続確認

(MPC-3000(SH2A) BL/I 2.00_01 2018/07/13)

- プログラム

```
/* 冒頭にEthernetの設定を記述します

SET_IP 192 168 0 20 255 255 255 0 192 168 0 248 /* 自分(MPC-3000)の設定
SET_IP TCP Modbus SRC 192 168 0 1                /* Epson RC の IP Address
/* SET_IP TCP Modbus SRC REF_T|20 192 168 0 1   /* 通信タイミングを指定する場合

*BASE0
 mm=0
  DO
    MBK(7004)=mm                                 /* Master(自分)のエリアに書き込む
    WAIT   MBK(7104)==mm :  mm=mm+1              /* Slave(RC)エリアの書き込み待ち
    IF   mm>60000 THEN  :  mm=0 : END_IF
    TIME   10
  LOOP
```

- 設定を有効化する手順

(1)MPCにプログラムを読み込みます。
(2)ダイレクトコマンドで　SET_IP VOID と実行します。
(3)プログラム実行(RUN)
(4)MPCの電源を再投入(パワーオンリセット)します。

この段階(まだRCと通信していない)では次のようになります。

```
++++++++++++++++++                    <<-- パワーオンリセット
Detect = 24AA02E48
IPadrs = 192.168.0.20                 <- 自己(MPC-3000)IP Address
 MASK  = 255.255.255.0
Gateway= 192.168.0.248
OFF Delay ACK
MACadrs= 001EC08DAB6E
Ether Opened
TCP_STACK_RAM=11656
TCP Init Completed
Wake TCP
  TASK 46 :  Modbus_master  SRC  ref_time=50  <- TASK46で Modbus通信
Wake UDP
Wake Serial
USB_I/F PIC24 Ver1.04
iMPC-3000
#
#SET_IP                                <<-- SET_IPコマンドで現在状況を表示します
IPadrs = 192.168.0.20
 MASK  = 255.255.255.0
Gateway= 192.168.0.248
OFF Delay ACK
MACadrs= 001EC08DAB6E

1 : /  TCP   Modbus       SRC     502     Master 192.168.0.1 ref_time = 50
#
```

- RCと通信を開始すると

```
#SET_IP
IPadrs = 192.168.0.20
 MASK  = 255.255.255.0
Gateway= 192.168.0.248
OFF Delay ACK
MACadrs= 001EC08DAB6E

1 : =  TCP   Modbus       SRC     502     Master 192.168.0.1 ref_time = 50
   ↑ ここが変る

1 : x  TCP   Modbus       SRC     502     Master 192.168.0.1 ref_time = 50
   ↑ エラー
```

■ MPC MBKエリアのビット操作

ビット入力　SW(mbk,bit)
ビットオン　ON (mbk,bit)
ビットオフ　OFF (mbk,bit)
　　↑ON/OFFの後はスペース1個空ける

※MBK(0)～MBK(10)のビット操作はできません。

・プログラム例

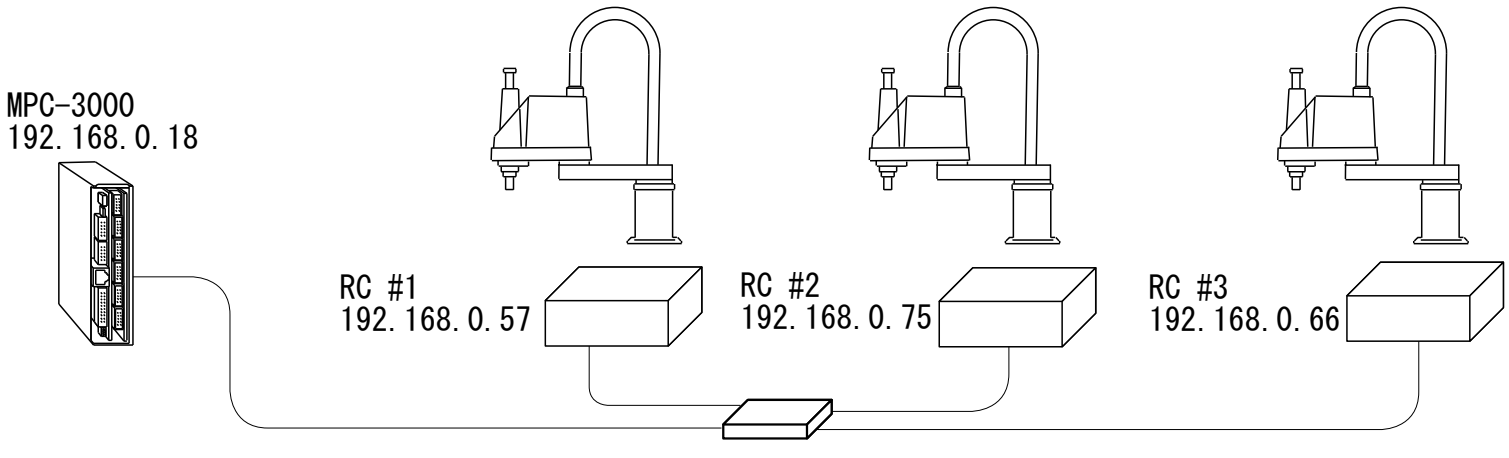
```
FOR i=0 TO 15
  ON (7000,i)
  PRINT "Bit" i "ON:" SW(7000,i) HEX$(MBK(7000))
NEXT
FOR i=0 TO 15
  OFF (7000,i)
  PRINT "Bit" i  "OFF:" SW(7000,i) HEX$(MBK(7000))
NEXT
```

・実行結果

```
RUN

# Bit 0 ON: 1 00000001
 Bit 1 ON: 1 00000003
 Bit 2 ON: 1 00000007
 Bit 3 ON: 1 0000000F
 Bit 4 ON: 1 0000001F
 Bit 5 ON: 1 0000003F
 Bit 6 ON: 1 0000007F
 Bit 7 ON: 1 000000FF
 Bit 8 ON: 1 000001FF
 Bit 9 ON: 1 000003FF
 Bit 10 ON: 1 000007FF
 Bit 11 ON: 1 00000FFF
 Bit 12 ON: 1 00001FFF
 Bit 13 ON: 1 00003FFF
 Bit 14 ON: 1 00007FFF
 Bit 15 ON: 1 0000FFFF
 Bit 0 OFF: 0 0000FFFE
 Bit 1 OFF: 0 0000FFFC
 Bit 2 OFF: 0 0000FFF8
 Bit 3 OFF: 0 0000FFF0
 Bit 4 OFF: 0 0000FFE0
 Bit 5 OFF: 0 0000FFC0
 Bit 6 OFF: 0 0000FF80
 Bit 7 OFF: 0 0000FF00
 Bit 8 OFF: 0 0000FE00
 Bit 9 OFF: 0 0000FC00
 Bit 10 OFF: 0 0000F800
 Bit 11 OFF: 0 0000F000
 Bit 12 OFF: 0 0000E000
 Bit 13 OFF: 0 0000C000
 Bit 14 OFF: 0 00008000
 Bit 15 OFF: 0 00000000
```

■ 最大3台のRCとメモリーシェアができます。

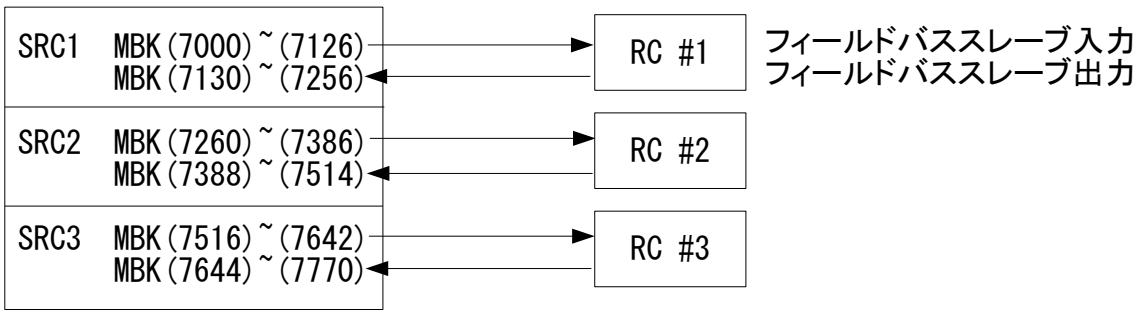
（サンプル作成: MPC-3000(SH2A) BL/I 2.01_25 2019/05/08）



MPCがMasterとなりModbus/TCPで各RCとメモリを共有。

■ MPC設定例

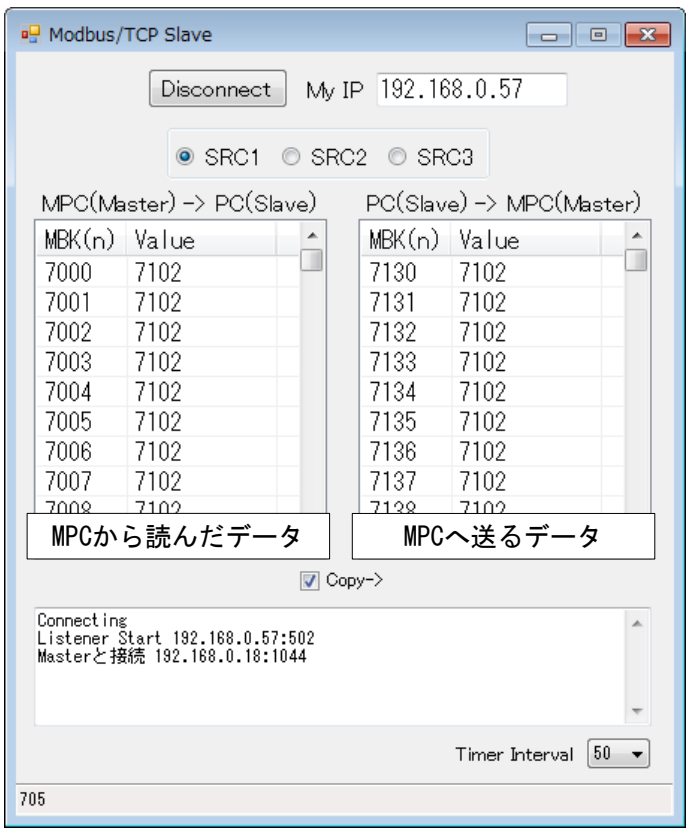
```
SET_IP 192 168 0 18 255 255 255 0 192 168 0 248
SET_IP TCP Modbus SRC1 REF_T|50 192 168 0 57
SET_IP TCP Modbus SRC2 REF_T|50 192 168 0 75
SET_IP TCP Modbus SRC3 REF_T|50 192 168 0 66
```

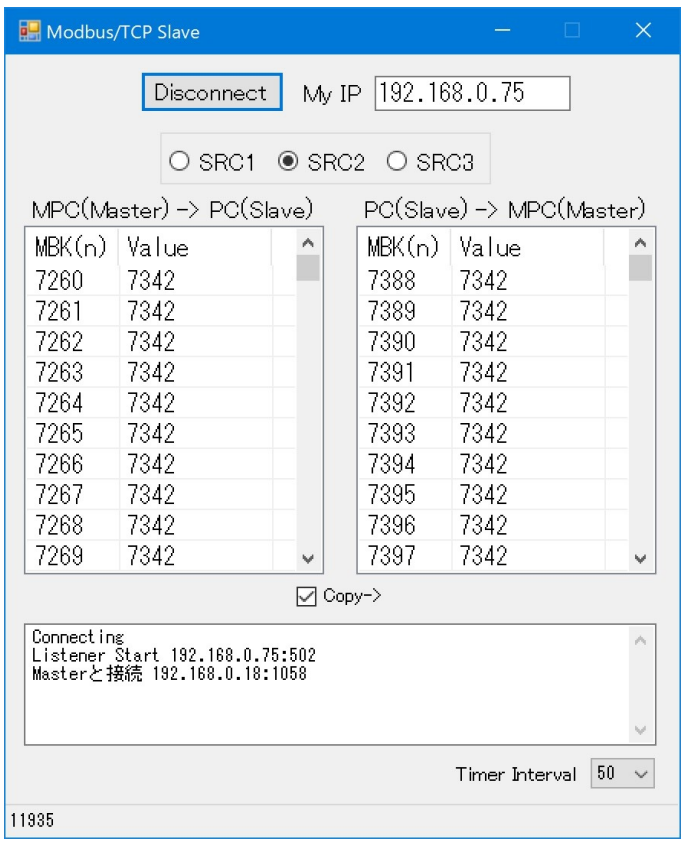


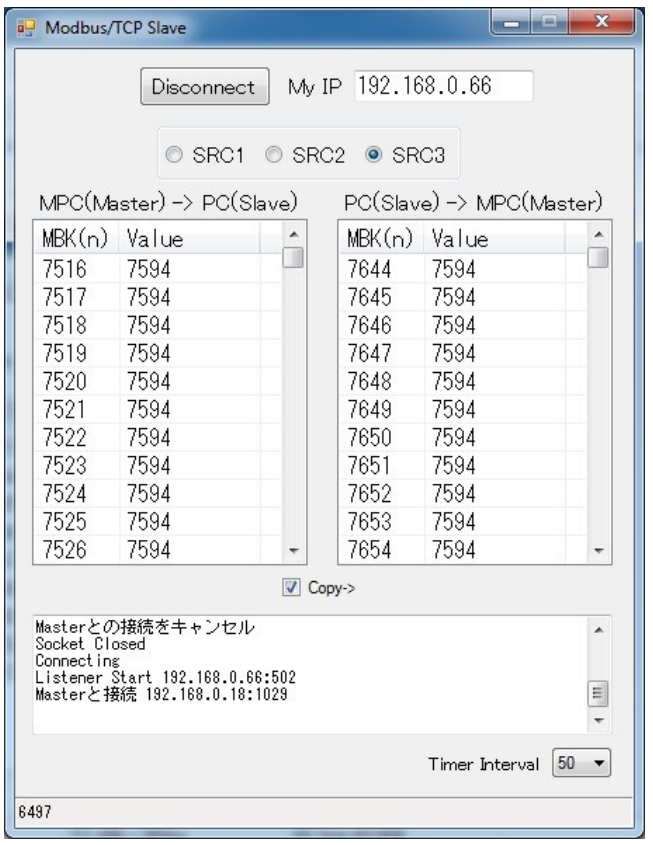
各MBKエリア 書込127word,読込127word
このエリアは固定です。

※ SRC と SRC1 は排他使用です。次のような使い方はできません。
```
SET_IP TCP Modbus SRC REF_T|50 192 168 0 57
SET_IP TCP Modbus SRC1 REF_T|50 192 168 0 75
```

- RCの代わりにパソコンのアプリとリンクすることもできます。
  次の3つは別々のパソコンで実行しています。

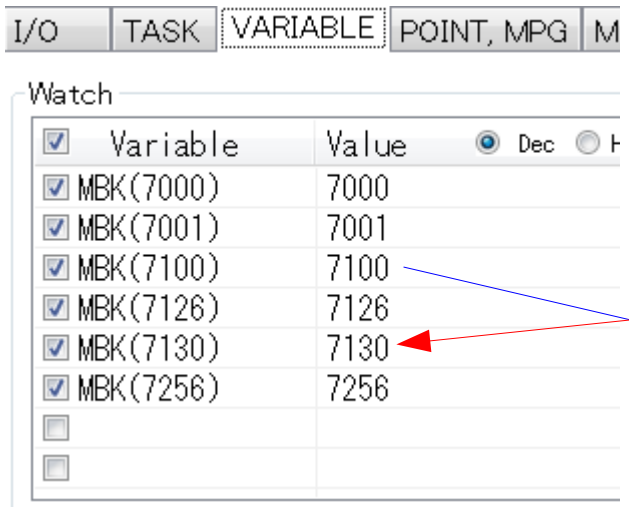






■ 通信内容のモニター方法
アプリケーションを作成するとき役立ちます。

MPCがMaster

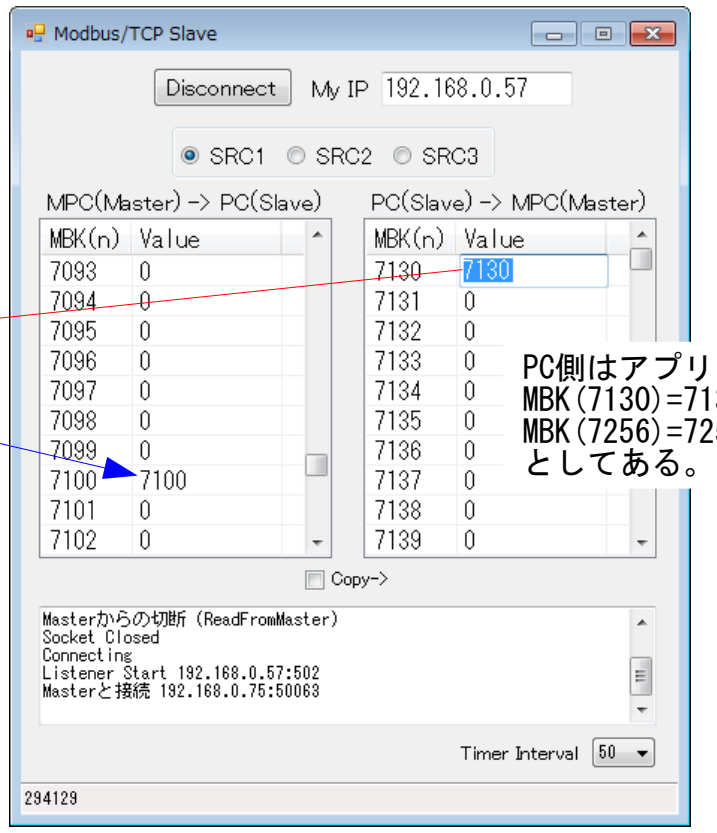


MPC側はMPC Monitorで
MBK(7000)=7000
MBK(7001)=7001
MBK(7100)=7100
MBK(7126)=7126
としてある。

Read

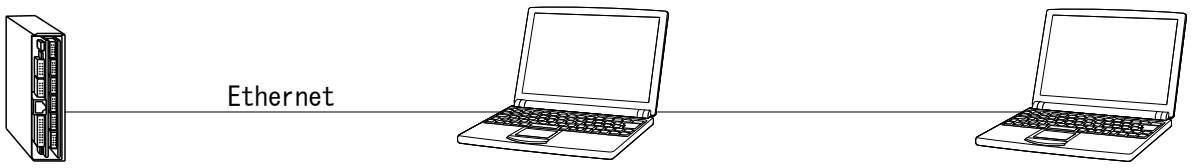
Write

Slaveアプリ

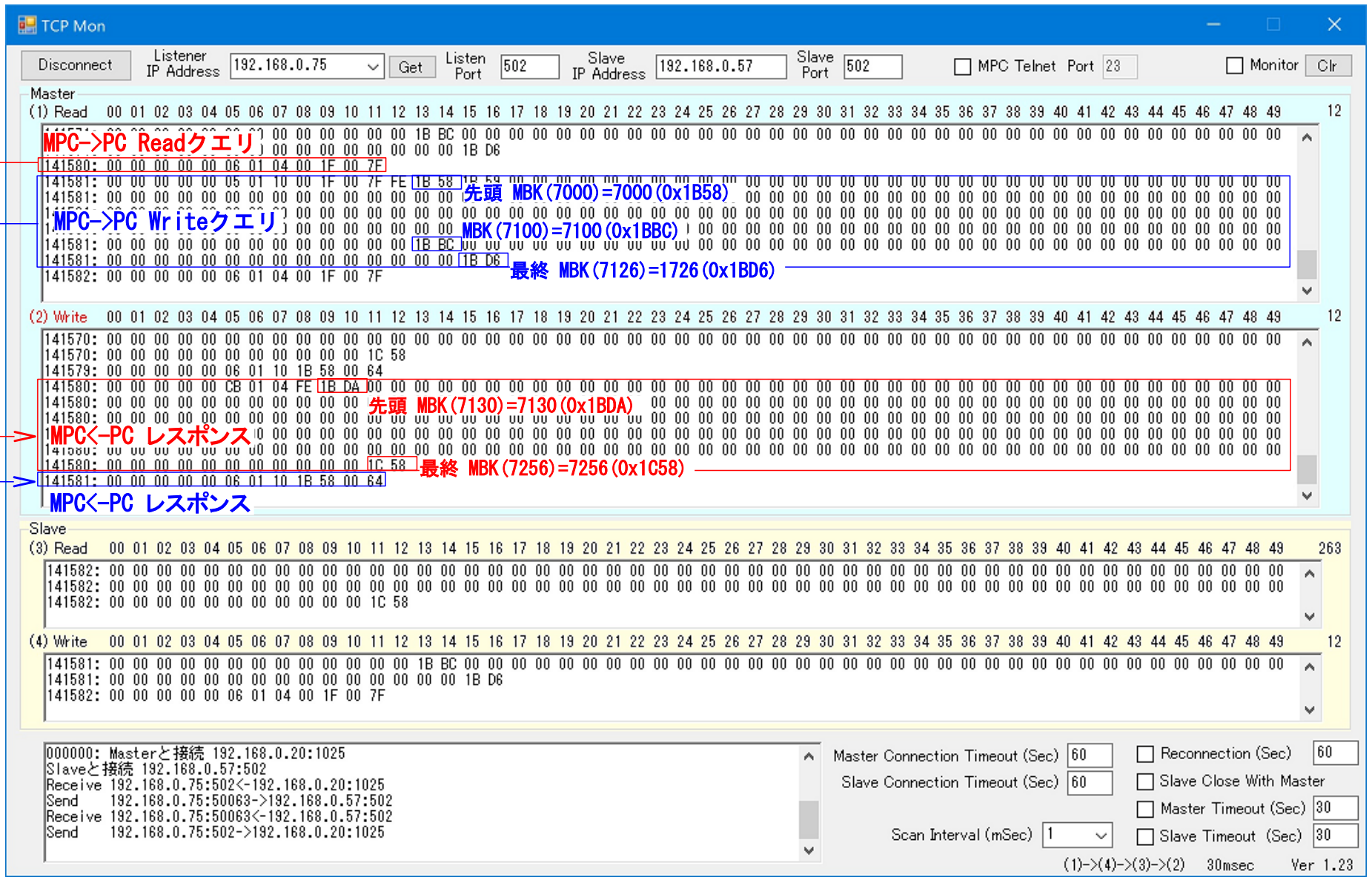


PC側はアプリで
MBK(7130)=7130
MBK(7256)=7256
としてある。

クエリ →
←レスポンス



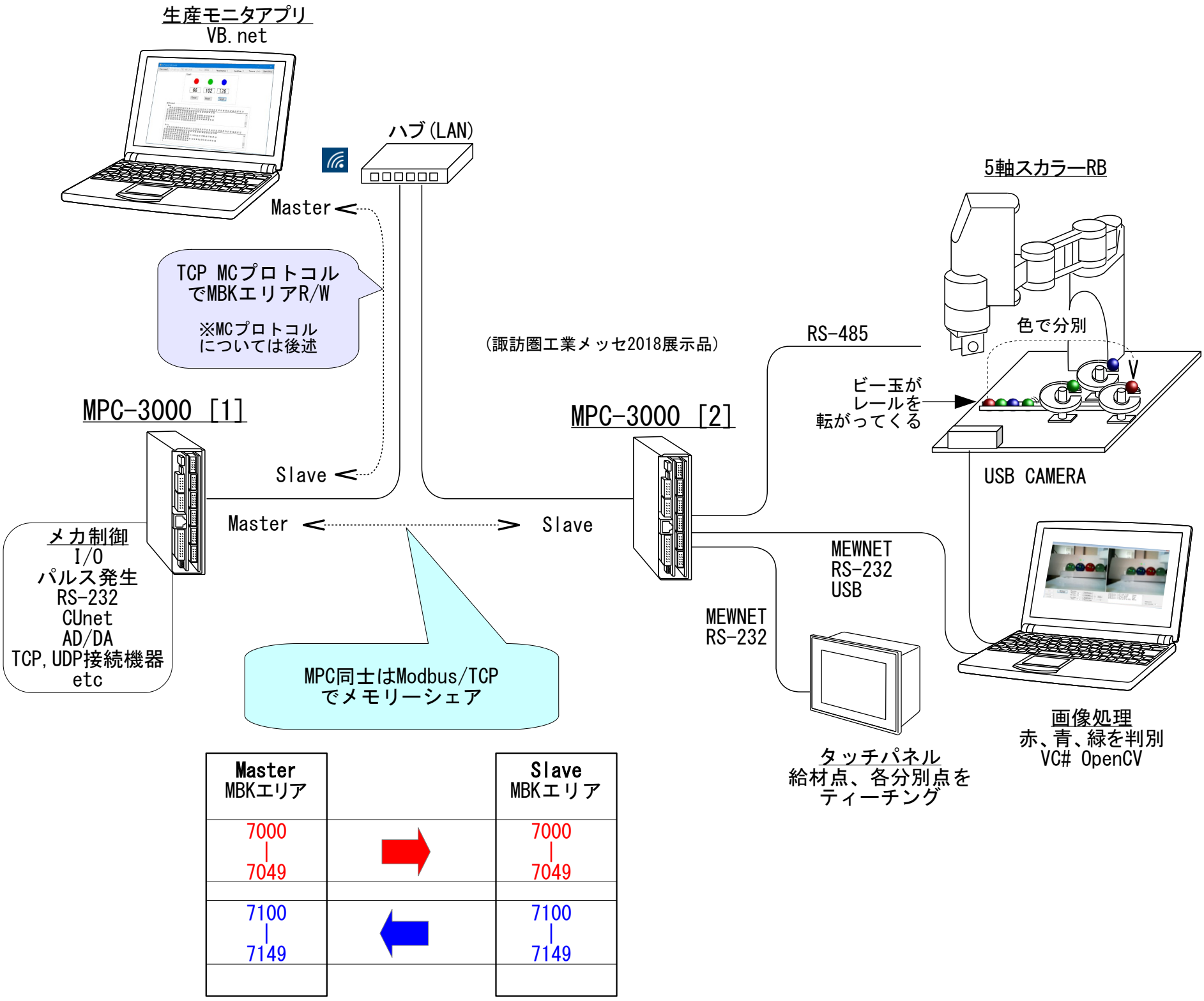
TCP MonでMPCとアプリ間のModbus/TCPをモニタ



MPC同士のメモリシェア

- サンプル作成: MPC-3000(SH2A) BL/I 2.00_10 2018/11/21

■ Modbus/TCP メモリシェア



Master MBKエリア		Slave MBKエリア
7000 | 7049	→	7000 | 7049
7100 | 7149	←	7100 | 7149

- Masterの設定

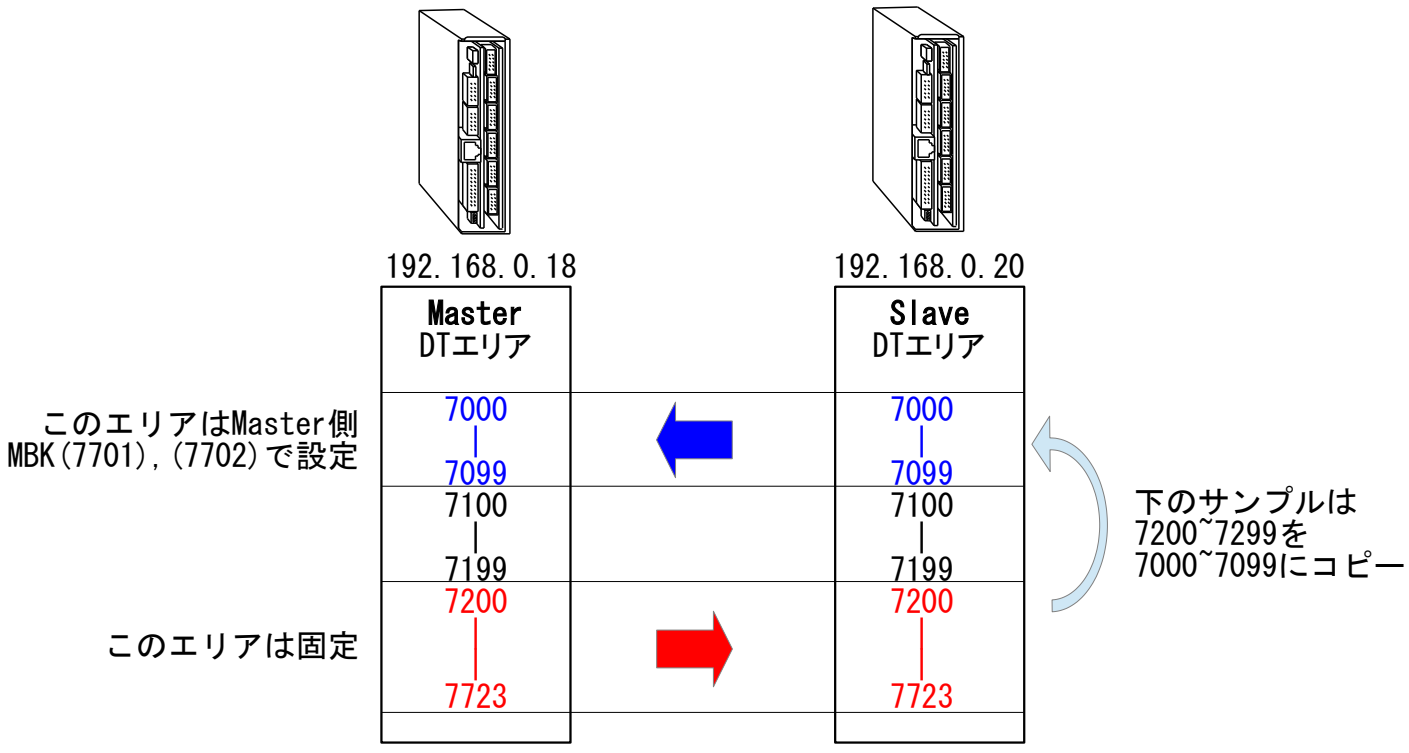
```
SET_IP 192 168 0 18 255 255 255 0 192 168 0 248 /* 自己IPアドレス設定
SET_IP TCP Modbus SRC REF_T|20 192 168 0 20      /* TCP Modbusマスタ設定
SET_IP TCP MC_PROTOCOL 10000                    /* TCP MCプロトコルスレーブ設定
```

- Slaveの設定

```
SET_IP 192 168 0 20 255 255 255 0 192 168 0 248 /* 自己IPアドレス設定
SET_IP TCP Modbus                               /* TCP Modbusスレーブ設定
SET_IP Serial MEWNET 1                          /* タッチパネル通信
SET_IP Serial MEWNET 3                          /* 画像処理PC通信
```

■ UDP メモリシェア 1:1

※ MBK() エリアは タッチパネル(MEWNET)と同じ表記 DT() でも記述できます。



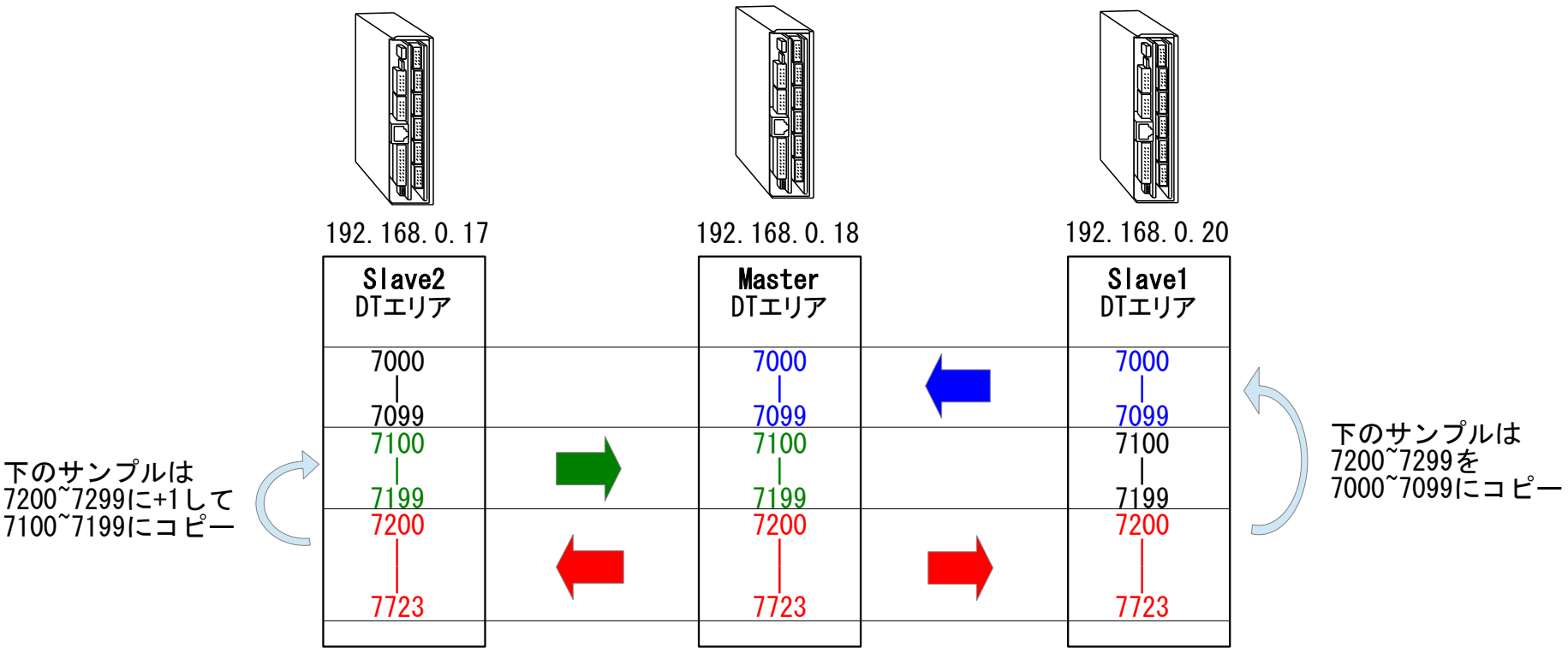
- Masterプログラム例

```
SET_IP 192 168 0 18 255 255 255 0 192 168 0 248   /* Own
SET_IP UDP SHARE 3000 REF_T|100 192 168 0 20      /* Slaveは 192.168.0.20
SET_IP TCP Telnet

MBK(7700)=20          /* SlaveのIP
MBK(7701)=7000        /* Slave読み出し先頭番号
MBK(7702)=100         /* データ数(Max499)
MBK(7703)=50          /* レスポンスディレイ mSec

DO
  FOR j=-32767 TO 32767
    FOR i=7200 TO (7200+DT(7702)-1)
      DT(i~Int)=j
    NEXT
    TIME 200
    FOR i=7000 TO (7000+DT(7702)-1)
      IF j<>DT(i~Int) THEN
        PR "Slave Mismatch"
        END
      END_IF
    NEXT
    PR "Slave Ok" j
  NEXT
LOOP
```

- Slaveプログラム例

```
SET_IP 192 168 0 20 255 255 255 0 192 168 0 248     /* Own
SET_IP UDP SHARE 3000                               /* Share Slave

DO
  DIMCPY DT(7200) DT(7000) 100
  SWAP
LOOP
```

■ UDP メモリシェア ブロードキャスト



- Masterプログラム例

```
SET_IP 192 168 0 18 255 255 255 0 192 168 0 248   /* Own
SET_IP UDP SHARE 3000 REF_T|100 255 255 255 255   /* Broadcast
SET_IP TCP Telnet

DT(7700)=20          /* Slave1のIP
DT(7701)=7000        /* Slave1読み出し先頭番号
DT(7702)=100         /* データ数
DT(7703)=5           /* レスポンスディレイ mSec

DT(7704)=17          /* Slave2のIP
DT(7705)=7100        /* Slave2読み出し先頭番号
DT(7706)=100         /* データ数
DT(7707)=10          /* レスポンスディレイ mSec

DO
  FOR j=-32767 TO 32767
    FOR i=7200 TO (7200+DT(7702)-1)
      DT(i~Int)=j
    NEXT
    TIME 200
    FOR i=7000 TO (7000+DT(7702)-1)
      IF j<>DT(i~Int) THEN
        PR "Slave1 Mismatch"
        END
      END_IF
    NEXT
    PR "Slave1 Ok" j
    FOR i=7100 TO (7100+DT(7702)-1)
      IF j<>(DT(i~Int)-1) THEN
        PR "Slave2 Mismatch"
        END
      END_IF
    NEXT
    PR "Slave2 Ok" j
  NEXT
LOOP
```

- Slave1プログラム例

```
SET_IP 192 168 0 20 255 255 255 0 192 168 0 248     /* Own
SET_IP UDP SHARE 3000                               /* Share Slave

DO
  DIMCPY DT(7200) DT(7000) 100
  SWAP
LOOP
```

- Slave2プログラム例

```
SET_IP 192 168 0 17 255 255 255 0 192 168 0 248     /* Own
SET_IP UDP SHARE 3000                               /* Share Slave

DO
  FOR i=7100 TO 7199
    DT(i)=DT(i+100)+1    /* Materから書き込まれたデータを自分のエリアにコピー
  NEXT
  SWAP
LOOP
```

MCプロトコル通信

■ TCP MCプロトコル(バイナリコード)による読み出し・書き込み （VB.netがマスター、MPCがスレーブ）

※MPC搭載のプロトコルは三菱MCプロトコル(3Eフレーム)に準じてMPC用にアレンジしたもので、完全互換ではありません。

・読み出し Function

要求伝文
senddat()

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
サブヘッダ		ネットワーク番号	PC番号	要求先ユニットI/O番号		要求先ユニット局番	送信データ長		CPU監視タイマー		コマンド		サブコマンド		デバイス先頭番号			デバイス	要求データ長	
50	00	00	FF	FF	03	00	0C	00	10	00	01	04	00	00	adr			dev	cnt	00

応答伝文
receivedat()

0	1	2	3	4	5	6	7	8	9	10	11	12	・・・
サブヘッダ		ネットワーク番号	PC番号	要求先ユニットI/O番号		要求先ユニット局番	応答データ長		終了コード		応答データ		応答データ
D0	00	00	FF	FF	03	00	下位	上位	00	00	下位	上位	下位 上位

```
Public Function Raed_Word(ByVal adr As Integer, ByVal dev As Integer, ByRef resarray() As Integer, ByVal tmout As Integer) As Integer

    Dim cnt As Integer = resarray.Length

    Try
        If MpcSck.Connected = False Then     ' Disconnect, FormClosing でオブジェクトが破棄されたとき
            MessageBox.Show("接続されていない", Application.ProductName)
            Return -3
        End If

        Dim senddat(20) As Byte
        senddat(0) = &H50 : senddat(1) = &H0                ' サブヘッダ
        senddat(2) = &H0 : senddat(3) = &HFF                ' ネットワーク番号
        senddat(4) = &HFF : senddat(5) = &H3 : senddat(6) = &H0 ' 二重化CPUでない
        senddat(7) = 12 : senddat(8) = &H0                  ' 送信データ長
        senddat(9) = &H10 : senddat(10) = &H0               ' CPU監視タイマー
        senddat(11) = &H1 : senddat(12) = &H4               ' コマンド
        senddat(13) = &H0 : senddat(14) = &H0               ' サブコマンド
        senddat(15) = adr And &HFF : senddat(16) = (adr >> 8) And &HFF : senddat(17) = (adr >> 16) And &HFF  ' デバイス先頭番号
        senddat(18) = dev                                   ' デバイスコード
        senddat(19) = cnt : senddat(20) = &H0               ' 要求データ長(ワード)

        MpcStm.Write(senddat, 0, senddat.GetLength(0))      ' TCP送信

        Dim Due As DateTime = Now.AddMilliseconds(CDbl(tmout))
        While MpcSck.Available = 0
            If Now > Due Then
                MessageBox.Show("Read Timeout")
                Return -1
            End If
            Sleep(1)
        End While

        Dim receivedat As Byte() = New Byte(MpcSck.Available - 1) {}
        MpcStm.Read(receivedat, 0, receivedat.GetLength(0))   ' TCP受信

        For i = 0 To cnt - 1
            resarray(i) = (CInt(receivedat(12 + i * 2)) << 8) + receivedat(11 + i * 2)
        Next
        Return 1

    Catch ex As Exception
        MessageBox.Show(ex.Message)
        Return -2
    End Try

End Function
```

・書き込みFunction

要求伝文
senddat()

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	･･･
サブヘッダ		ネットワーク番号	PC番号	要求先ユニットI/O番号		要求先ユニット局番	送信データ長		CPU監視タイマー		コマンド		サブコマンド		デバイス先頭番号			デバイス	書込データ長		書込データ	
50	00	00	FF	FF	03	00	0C	00	10	00	01	14	00	00	adr			dev	cnt	00	wdat()	

応答伝文
receivedat()

0	1	2	3	4	5	6	7	8	9	10
サブヘッダ		ネットワーク番号	PC番号	要求先ユニットI/O番号		要求先ユニット局番	応答データ長		終了コード	
D0	00	00	FF	FF	03	00	02	00	00	00

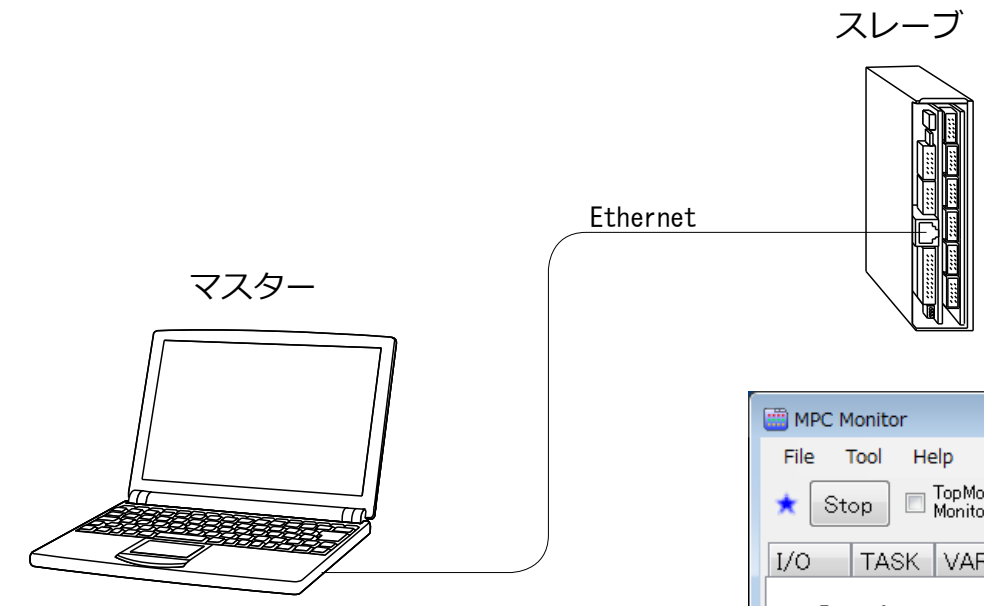
```
Public Function Write_Word(ByVal adr As Integer, ByVal dev As Integer, ByVal writearray() As Integer, ByVal tmout As Integer) As Integer

    Dim cnt As Integer = writearray.Length

    Try
        If MpcSck.Connected = False Then    ' Disconnect, FormClosing でオブジェクトが破棄されたとき
            MessageBox.Show("接続されていない", Application.ProductName)
            Return -3
        End If
        Dim wdat(cnt * 2 - 1) As Byte
        For i = 0 To cnt - 1
            wdat(i * 2) = writearray(i) And &HFF
            wdat(i * 2 + 1) = (writearray(i) >> 8) And &HFF
        Next

        Dim senddat(wdat.Length + 20) As Byte
        senddat(0) = &H50 : senddat(1) = &H0                ' サブヘッダ
        senddat(2) = &H0 : senddat(3) = &HFF                ' ネットワーク番号
        senddat(4) = &HFF : senddat(5) = &H3 : senddat(6) = &H0 ' 二重化CPUでない
        senddat(7) = 12 + wdat.Length : senddat(8) = &H0    ' 送信データ長
        senddat(9) = &H10 : senddat(10) = &H0               ' CPU監視タイマー
        senddat(11) = &H1 : senddat(12) = &H14              ' コマンド
        senddat(13) = &H0 : senddat(14) = &H0               ' サブコマンド
        senddat(15) = adr And &HFF : senddat(16) = (adr >> 8) And &HFF : senddat(17) = (adr >> 16) And &HFF  ' デバイス先頭番号
        senddat(18) = dev                                   ' デバイスコード
        senddat(19) = cnt : senddat(20) = &H0               ' 書込データ長(ワード)
        Array.Copy(wdat, 0, senddat, 21, wdat.Length)

        MpcStm.Write(senddat, 0, senddat.GetLength(0))      ' TCP送信

        Dim Due As DateTime = Now.AddMilliseconds(CDbl(tmout))
        While MpcSck.Available = 0
            If Now > Due Then
                MessageBox.Show("Write Timeout")
                Return -1
            End If
            Sleep(1)
        End While

        Dim receivedat As Byte() = New Byte(MpcSck.Available - 1) {}
        MpcStm.Read(receivedat, 0, receivedat.GetLength(0)) 'TCP受信

        Return 1
    Catch ex As Exception
        MessageBox.Show(ex.Message)
        Return -2
    End Try

End Function
```

・デバイス定義

```
Public Enum Device
    devD = &HA8    'MPCのMBKエリア
    devW = &HB4
    devR = &HAF
    devM = &H90
    devX = &H9C
    devY = &H9D
    devB = &HA0
    devL = &H92
End Enum
```

■ MCプロトコルアプリケーション実行例

前述の読み出し、書き込みFunctionを使って作成したアプリケーションです。

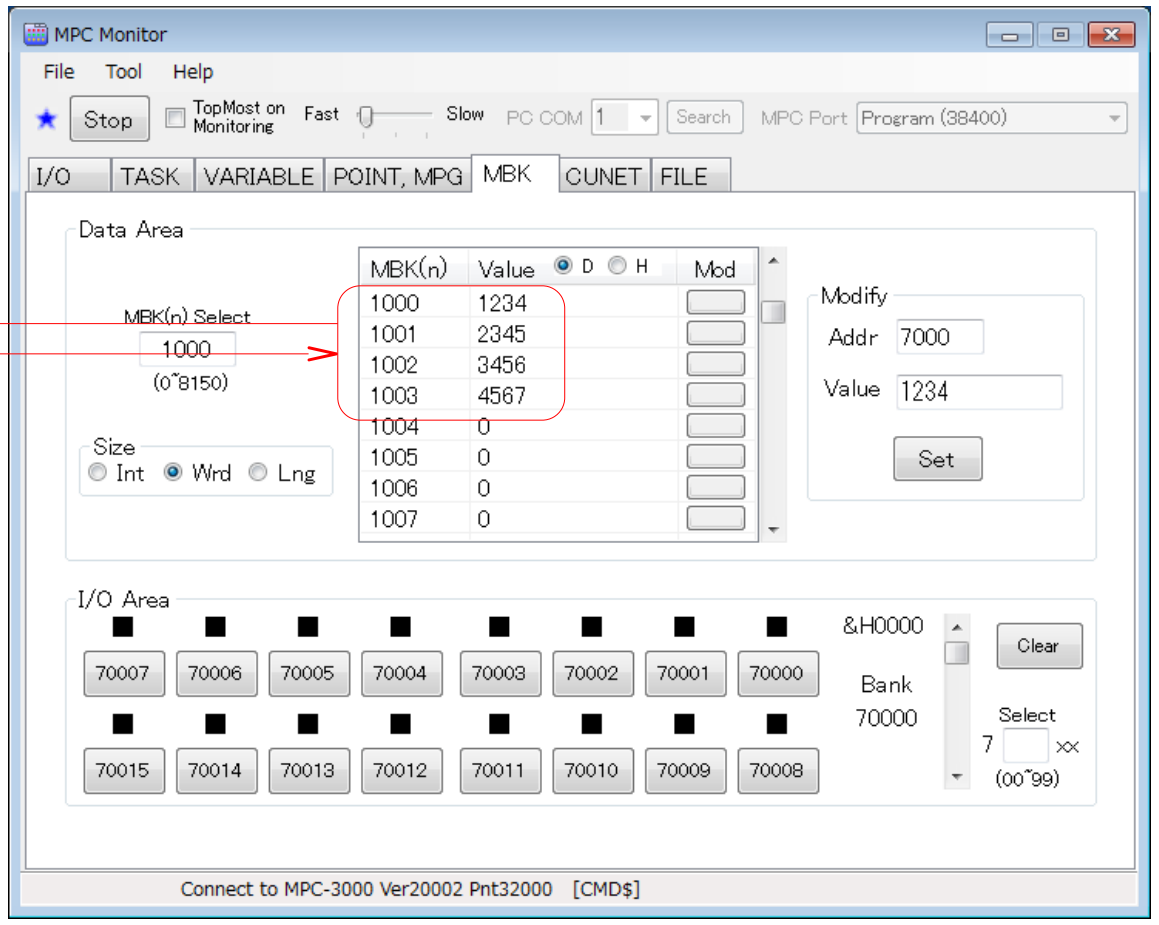


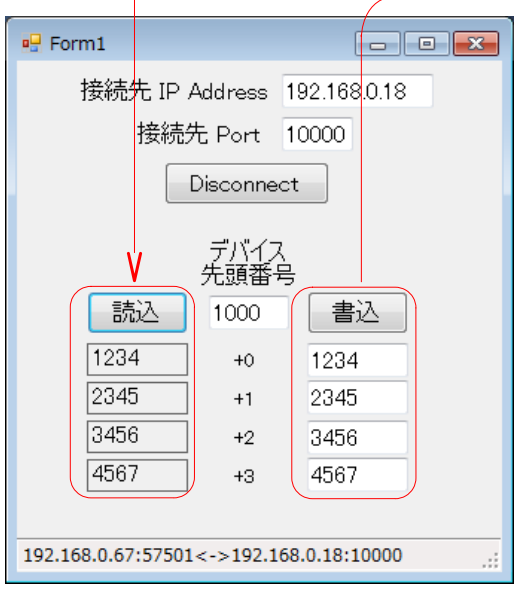
・MPCの宣言

```
SET_IP TCP MC_PROTOCOL 10000
```

（プロトコルはバックグラウンドで実行されるため、
MPCはプログラムが空でも実行できます）

MPC Monitor

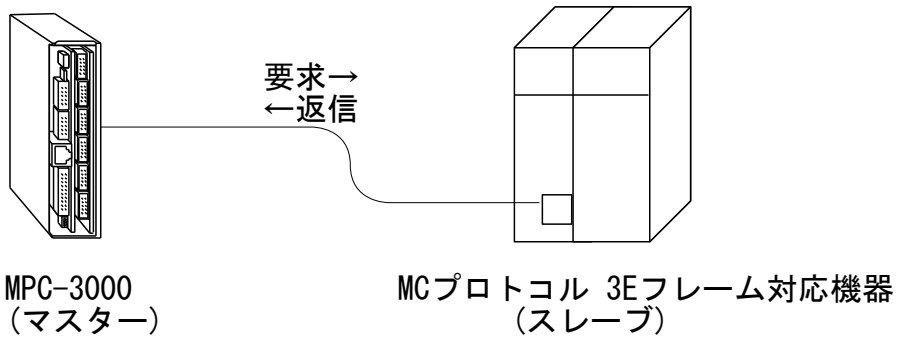




・ボタンクリックイベント

```
    ''' <summary>
    ''' 読込ボタン
    ''' </summary>
    Private Sub Button_Read_Click(sender As Object, e As EventArgs) Handles Button_Read.Click
        Dim resarray(3) As Integer
        McProtocol.Raed_Word(TextBox_DevHead.Text, McProtocol.Device.devD, resarray, 2000)
        Label_Read0.Text = resarray(0)
        Label_Read1.Text = resarray(1)
        Label_Read2.Text = resarray(2)
        Label_Read3.Text = resarray(3)

    End Sub

    ''' <summary>
    ''' 書込ボタン
    ''' </summary>
    Private Sub Button_Write_Click(sender As Object, e As EventArgs) Handles Button_Write.Click
        Dim writedat() As Integer = {TextBox_Write0.Text, TextBox_Write1.Text, TextBox_Write2.Text, TextBox_Write3.Text}
        McProtocol.Write_Word(TextBox_DevHead.Text, McProtocol.Device.devD, writedat, 2000)

    End Sub
```

■ TCP MCプロトコル(バイナリコード)による読み出し・書き込み (MPCがマスター、接続機器がスレーブ)



MPCがマスターになる場合、PACKET 通信の Q3E コマンドを用います。

・MPCサンプルプログラム

```
SET_IP 192 168 0 20 255 255 255 0 192 168 0 248
SET_IP TCP PACKET 9000 192 168 0 67   /* Master(MPC) -> Slave 192,168,0,67:9000

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 "D" 10000 X(10000) 4 /* X(10000)～X(10003)をSlaveのD10000～D10003にコピー
/* 通信内容
/*  MPC->Slave 要求
/*  50 00 00 FF FF 03 00 14 00 10 00 01 14 00 00 10 27 00 A8 04 00 10 27 11 27 12 27 13 27
/*  Slave->MPC 応答
/*  D0 00 00 FF FF 03 00 02 00 00 00
PRINT "OK"

PRINT "READ_BULK"
Q3E TCP|0 READ_BULK "D" 10000 Y(10000) 4  /* SlaveのD10000～D10003をY(10000)～Y(10003)にコピー
/* 通信内容
/*  MPC->Slave 要求
/*  50 00 00 FF FF 03 00 0C 00 10 00 01 04 00 00 10 27 00 A8 04 00
/*  Slave->MPC 応答
/*  D0 00 00 FF FF 03 00 0A 00 00 00 10 27 11 27 12 27 13 27
FOR i=10000 TO 10003
  PRINT "P(" i ")" P(i)
NEXT

PRINT "WRITE_RNDM"
Q3E  TCP|0 WRITE_RNDM "D" 10000 1234 "D" 10001 5678 "M" 100 8888 /* SlaveのD10000=1234,D10001=5678,M100=8888
/* 通信内容
/*  MPC->Slave 要求
/*  50 00 00 FF FF 03 00 1A 00 10 00 02 14 00 00 03 00 10 27 00 A8 D2 04 11 27 00 A8 2E 16 64 00 00 90 B8 22
/*  Slave->MPC 応答
/*  D0 00 00 FF FF 03 00 02 00 00 00
PRINT "OK"

PRINT "READ_RNDM"
Q3E  TCP|0 READ_RNDM "D" 10000 "D" 10001 "M" 100 MBK(100) /* SlaveのD10000,D10001,M100をMBK(100)～MBK(102)にコピー
/* 通信内容
/*  MPC->Slave 要求
/*  50 00 00 FF FF 03 00 14 00 10 00 03 04 00 00 03 00 10 27 00 A8 11 27 00 A8 64 00 00 90
/*  Slave->MPC 応答
/*  D0 00 00 FF FF 03 00 08 00 00 00 D2 04 2E 16 B8 22

FOR i=100 TO 102
  PRINT "MBK(" i ")" MBK(i)
NEXT
```

・SET_IP確認

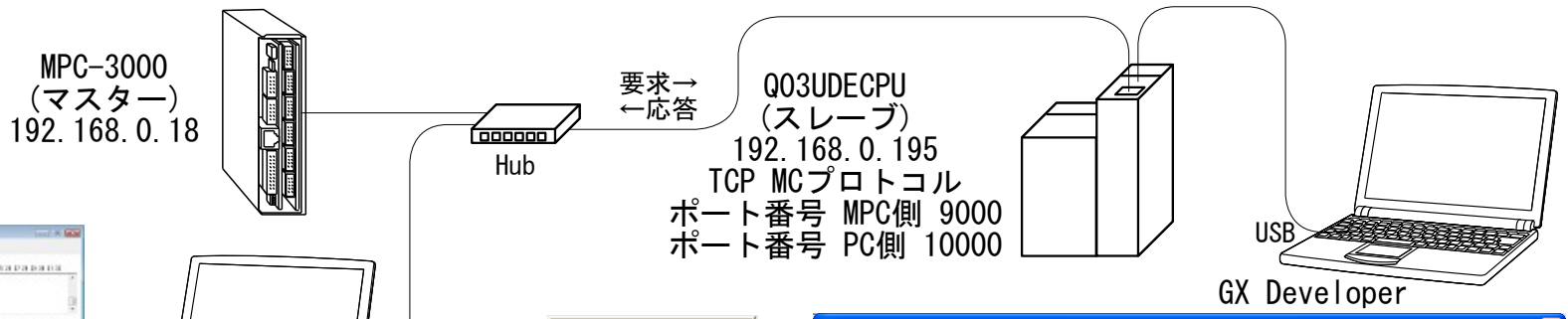
```
1 : =  TCP PACKET 9000 <TCP_XO$><TCP_RO$> Master 192.168.0.67 ref_time = 50
                            ↑TCP|ch の番号
```

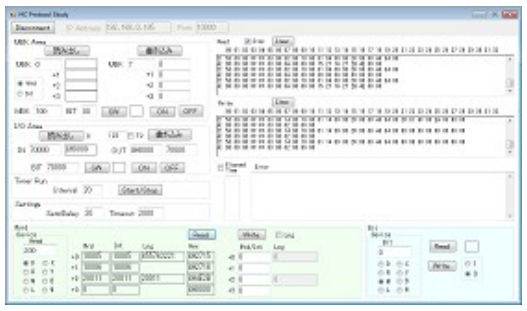
・実行結果

```
RUN

# WRITE_BULK
 OK
 READ_BULK
 P( 10000 ) 10000 10000 0 0
 P( 10001 ) 10001 10001 0 0
 P( 10002 ) 10002 10002 0 0
 P( 10003 ) 10003 10003 0 0
 WRITE_RNDM
 OK
 READ_RNDM
 MBK( 100 ) 1234
 MBK( 101 ) 5678
 MBK( 102 ) 8888
```

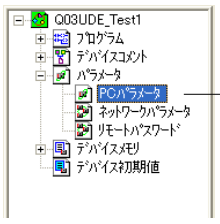
三菱シーケンサQ03UDECPUとの通信

■ TCP通信例

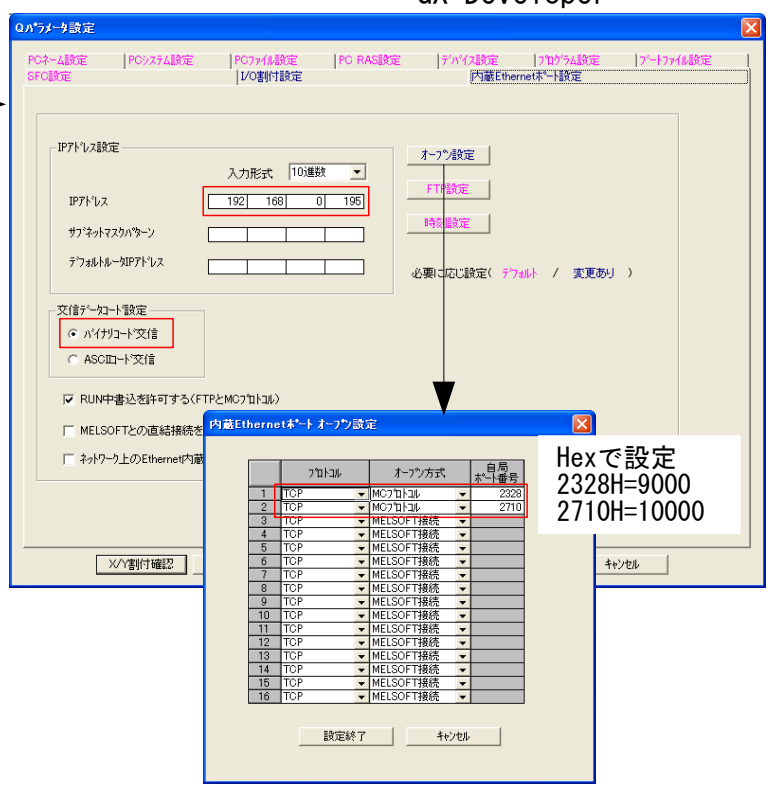




Q03デバイスの読み書き自作アプリ



プロジェクトデーター覧



Hexで設定
2328H=9000
2710H=10000

- MPCプログラム例1

```
/* (1)MPCのX(200),X(201)をQ03のD200,D201に書き込みます。
/*    Q03には D202=D200+D201 とプログラムしてあります。
/* (2)Q03のD200~D202をMPCのY(200)~Y(202)に読み込みます。

SET_IP 192 168 0 18
SET_IP IP_MASK 255 255 255 0
SET_IP IP_GATE 0 0 0 0
SET_IP TCP PACKET 9000 192 168 0 195        /* Q03UDECPUがスレーブ

FOR i=10000 TO 10005
  X(200)=i
  X(201)=i+1
  Q3E TCP|0 WRITE_BULK "D" 200 X(200) 2     /* X(200)~X(201) -> Q03 D200~D201
  Q3E TCP|0 READ_BULK "D" 200 Y(200) 3      /* Y(200)~Y(202) <- Q03 D200~D202
  PR "(D200)=Y(200) "+(D201)=Y(201) "=(D202)=Y(202)
NEXT
```

GX Developer デバイス一括モニタ

デバイス	+FEDC +BA98 +7654 +3210	
D200	○○●○ ○●●● ○○○● ○●○●	10005
D201	○○●○ ○●●● ○○○● ○●●○	10006
D202	○●○○ ●●●○ ○○●○ ●○●●	20011

- 実行結果

```
RUN

# (D200) 10000 +(D201) 10001 =(D202) 20001
  (D200) 10001 +(D201) 10002 =(D202) 20003
  (D200) 10002 +(D201) 10003 =(D202) 20005
  (D200) 10003 +(D201) 10004 =(D202) 20007
  (D200) 10004 +(D201) 10005 =(D202) 20009
  (D200) 10005 +(D201) 10006 =(D202) 20011
```

```
MPCのチャンネル番号はSET_IPコマンドで確認できます。

#SET_IP
IPadrs = 192.168.0.18
 MASK  = 255.255.255.0
Gateway= 0.0.0.0
OFF Delay ACK  baud=115200
MACadrs= 001EC08DAB6E

1 : /  TCP   PACKET  9000 <TCP_XO$><TCP_RO$> Master 192.168.0.195 ref_time = 50
```

- MPCプログラム例2

```
/* (1)PCアプリでQ03のD1000～1099にデータを書き込みます。
/* (2)PCアプリでY0をONします。MPCはそれをトリガとしてD1000～1099をX(1000)～(1099)にコピーします。
/* (3)MPCはY0==0を待ちます。

SET_IP 192 168 0 18
SET_IP IP_MASK 255 255 255 0
SET_IP IP_GATE 0 0 0 0
SET_IP TCP PACKET 9000 192 168 0 195          /* Q03UDECPUがスレーブ
DO                                            /* PCアプリがY0をオンするのを待つ
  Q3E TCP|0 READ_RNDM "Y" 0 MBK(0)
  IF MBK(0)&&h1==1 THEN : BREAK : END_IF
LOOP
Q3E TCP|0 READ_BULK "D" 1000 X(1000) 100    /* X(1000)~X(1099) <- Q03 D1000~D1099
FOR p=1000 TO 1003 : PR "(D" p ")=X(" p ")==" X(p) : NEXT
FOR p=1096 TO 1099 : PR "(D" p ")=X(" p ")==" X(p) : NEXT
PR "Wait Start Bit==0"
DO                                            /* PCアプリがY0をオフするのを待つ
  Q3E TCP|0 READ_RNDM "Y" 0 MBK(0)
  IF MBK(0)&&h1==0 THEN : BREAK : END_IF
LOOP
LOOP
```

- 実行結果

```
RUN

# Wait Start Bit==1
 (D 1000 )=X( 1000 )== 1111  ← PCでD1000に1111を書き込んでおいた
 (D 1001 )=X( 1001 )== 2222
 (D 1002 )=X( 1002 )== 3333
 (D 1003 )=X( 1003 )== 4444
 (D 1096 )=X( 1096 )== 5555  ← PCでD1096に5555を書き込んでおいた
 (D 1097 )=X( 1097 )== 6666
 (D 1098 )=X( 1098 )== 7777
 (D 1099 )=X( 1099 )== 8888
Wait Start Bit==0
Wait Start Bit==1
```

デバイス	+FEDC +BA98 +7654 +3210	
D1000	○○○○ ○●○○ ○●○● ○●●●	1111
D1001	○○○○ ●○○○ ●○●○ ●●●○	2222
D1002	○○○○ ●●○● ○○○○ ○●○●	3333
D1003	○○○● ○○○● ○●○● ●●○○	4444

デバイス	+FEDC +BA98 +7654 +3210	
D1096	○○○● ○○○● ○●●● ○○●●	5555
D1097	○○○● ●○●○ ○○○○ ●○●○	6666
D1098	○○○● ●●●○ ○●●○ ○○○●	7777
D1099	○○●○ ○○●○ ●○●● ●○○○	8888

■ ビット書き込み（サポート MPC-3000(SH2A) BL/I 2.01_30 2019/07/10 以降）
MBK()等の配列を介さず直接ビットデバイスを操作します。（ビット読込はありません。）

- 書式

```
Q3E TCP|ch WRITE_BIT "M" head count data {data ..}
      ch   : チャンネル番号
      "M"  : デバイスM(内部リレー)
      head : 先頭番号
      count: ビット数
      data : 1=ON, 0=OFF
```

- 使い方

```
Q3E TCP|0 WRITE_BIT "M" 1 1 1            /* M1から1ビットを1にする。つまり M1=1
Q3E TCP|0 WRITE_BIT "M" 1000 4 1 1 1 1   /* M1000=1,M1001=1,M1002=1,M1003=1
```

- プログラム例

```
MBK(1000~Lng)=0
Q3E TCP|0 WRITE_BULK "M" 1000 MBK(1000) 2 /* 使う範囲を初期化

DO
  proc=1
  Q3E TCP|0 WRITE_BIT "M" 1000 1 0         /* M1000=0
  Q3E TCP|0 WRITE_BIT "M" 1016 4 1 0 1 0   /* M1016=1, M1017=0, M1018=1, M1019=0
  Q3E TCP|0 READ_BULK "M" 1000 MBK(1000) 2 /* M1000～32bit -> MBK(1000~Lng)
  PR proc HEX$(MBK(1000~Lng))
  TIME 500
  proc=2
  Q3E TCP|0 WRITE_BIT "M" 1000 1 1         /* M1000=1
  Q3E TCP|0 WRITE_BIT "M" 1016 4 0 1 0 1   /* M1016=0, M1017=1, M1018=0, M1019=1
  Q3E TCP|0 READ_BULK "M" 1000 MBK(1000) 2
  PR proc HEX$(MBK(1000~Lng))
  TIME 500
LOOP
```

- 実行結果

```
 1 00050000
 2 000A0001
 1 00050000
 2 000A0001
```

GX Developer デバイス モニタ

ﾃﾞﾊﾞｲｽ	+1 FEDC BA98 7654 3210	+0 FEDC BA98 7654 3210
M1000	0000 0000 0000 0101	0000 0000 0000 0000

ﾃﾞﾊﾞｲｽ	+1 FEDC BA98 7654 3210	+0 FEDC BA98 7654 3210
M1000	0000 0000 0000 1010	0000 0000 0000 0001

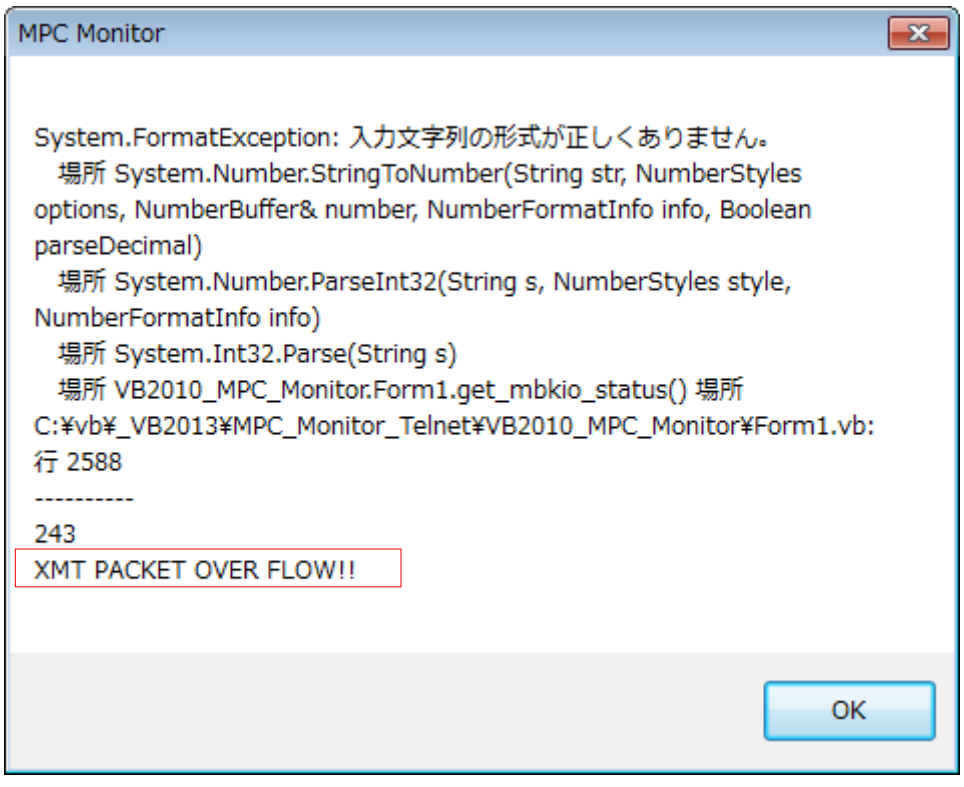
■ 複数のタスクからのアクセス
Q3E等のPACKETコマンドで複数のタスクから同じ相手にアクセスする場合は衝突しないように気をつけてください。
基本的には単一のタスクで通信し、複数のタスクで行う場合はセマフォしてください。

■ 実行時のエラー
BULKで読み書きする数が多いとパケットサイズも大きくなります。パケットサイズは512byteに限定されており、それを超えるとエラーになります。200ワード程度としてください。

例)書き込み時のエラー MPC-3000(SH2A) BL/I 2.01_30 2019/06/27 では 241ワード以上でエラー

```
  #Q3E TCP|0 WRITE_BULK "M" 1568 MBK(1680) 241
  XMT PACKET OVER FLOW!!
```

MPC MonitorのFILEタブ以外では次のようなメッセージが表示されることがあります(スキャンタイミングにより表示されないこともあります)



■ Q3E コマンドにタイムアウトオプションを追加(MPC-3000(SH2A) BL/I 2.01_48 2021/02/17 以降)。

オプション無し タイムアウト 10 秒　　Q3E TCP|p WRITE_BULK "D" 1000 aho(50) 3
タイムアウト　n=1～30 [秒]　　Q3E (n,TCP|p) WRITE_BULK "D" 1000 aho(50) 3

エラーコード
88　「TCP/IPが切断されています」
90　「TCP_XMTタイムアウトエラー」
91　「TCP_RCVタイムアウトエラー」

■ QO3UDECPU と UDP通信

TCPに代えてUDPでMCプロトコル通信を行います。
UDPはポートをクローズせずにMPCまたはQO3UDECPUの電源を切っても、再投入で直ちに通信が回復します。
データの消失を前提とし、タイムアウト処理で再送信を行います(...と言ってもTCPの場合と同じです)。
通信の負荷でタイムアウトの頻度は変わります。
(サンプル作成 BL/I 2.01_49 2021/02/25)

- GX Developerの設定

TCPをUDPに変更

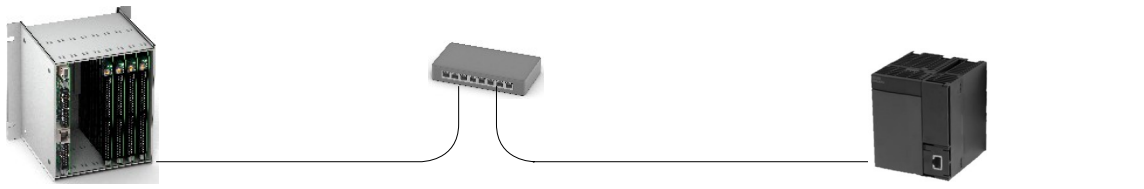
- MPCプログラム例

```
SET_IP TCP Telnet
SET_IP 192 168 0 18 255 255 255 0 192 168 0 248 /* 自己IP
SET_IP Serial MEWNET 2 38400                    /* タッチパネル宣言

/* UDP通信設定
SET_IP UDP PACKET 11000                         /* QO3UDECPU のPort番号
IP_CONV 192 168 0 195 DestIP                    /* 変数 DestIP に QO3UDECPU の IP を入れる
SET_DEST DestIP UDP_XO$                         /* 宛先設定

errmsg$=""
ON_ERROR  *on_err                               /* タイムアウトエラー処理

DO
  rw=100
  Q3E (3,UDP|0) WRITE_BULK "D" 500 X(200) rw    /* X(200)~rw word -> QO3 D200~
  TIME 50
  Q3E (3,UDP|0) READ_BULK "D" 200 Y(200) rw     /* Y(200)~ <- QO3 D200~rw word
  TIME 50
LOOP

*on_err
  errmsg$=TIME$(1)+" Line:"+STR$(err_&&H00FFFFFF)+" Code:"+STR$(err_>>24)+" "+ERR$(err_)
  PR errmsg$
  RESUME                                        /* エラー発生行を再実行
  END
```

- エラーメッセージ errmsg$ 例（エラーの文字列は「TCP」となっています）

15:24:12 Line:130 Code:91 TCP_RCVタイムアウトエラー

■ Q03UDECPU 再接続について(TCPの場合)



- MPCの電源をOFFする場合はその前に通信をCloseします。MPCの電源を入れ直せば再接続します。

```
#SET_IP
IPadrs = 192.168.0.18
 MASK  = 255.255.255.0
Gateway= 0.0.0.0
OFF Delay ACK  baud=38400
MACadrs= 001EC08DAB6E

1 : =  TCP   PACKET        9000     <TCP_XO$><TCP_RO$> Master 192.168.0.195 ref_time = 50

#PR TCP(0)
 1
#a=TCP(-1,0)
#PR TCP(0)
 2
#
MKY =0
+++++++++++++++++
Detect = 24AA02E48
IPadrs = 192.168.0.18
(中略)
iMPC-3000
#PR TCP(0)
 1
#Q3E TCP|0 WRITE_BIT "M" 1000 1 1
#
```

ch番号

現在接続中

Ch0をCloseする

ここでMPC電源OFF/ON

PLC側は何もしない

自動的に接続

コマンドも通る

- PLCの電源をOFF/ONしたときの再接続例

```
#PR TCP(0)
 1

#PR TCP(0)
 1

#a=TCP(-1,0)
#a=TCP(-2,0)

#PR TCP(0)
 1
#Q3E TCP|0 WRITE_BIT "M" 1000 1 1
#
```

PLCの電源を切る

直ぐには0にならない
(0になることもある)

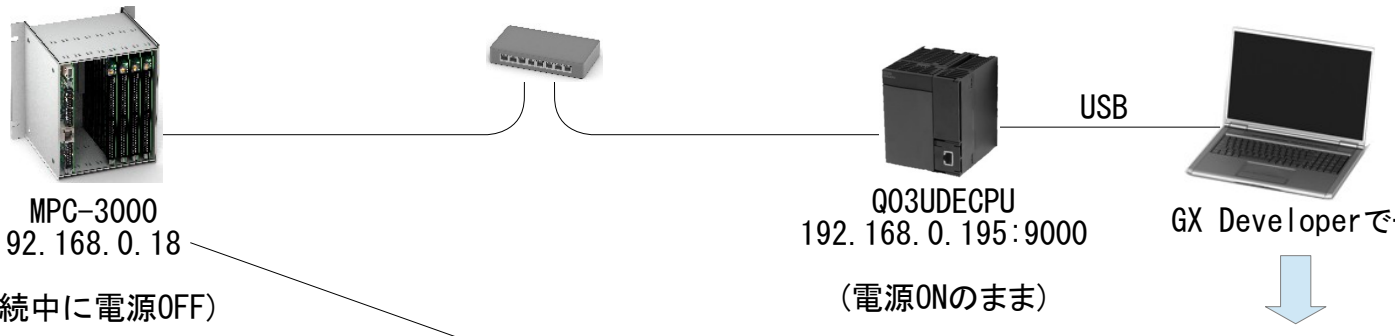
PLCの電源を入れる

Close
Reopen

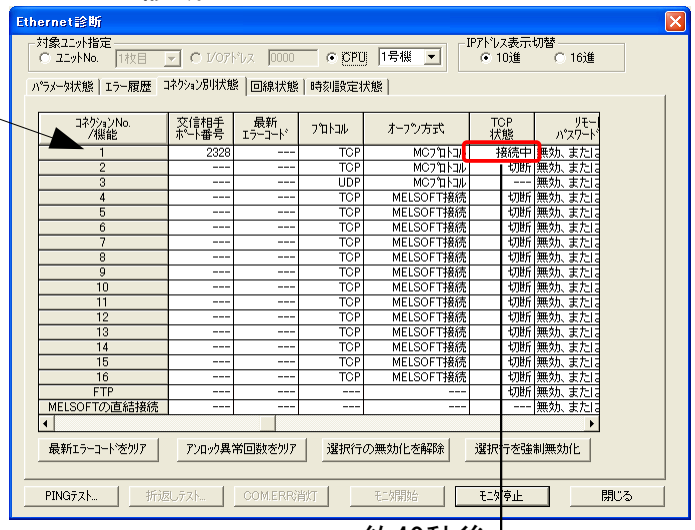
復旧するのに時間がかかることがある

コマンドが通る

- MPCをCloseせずに電源OFFしたときは、40秒以上経過後電源ONします。



Ethernet診断



(a)TCP接続中

MPC側の確認
#PR TCP(0)
1

(b)Close処理をせずにMPC電源OFF。

(c)OFFしたまま約40秒経つとPLC側が「切断」になります。

(d)MPC電源ONで再び「接続中」となる。(接続処理で数秒かかることがある)

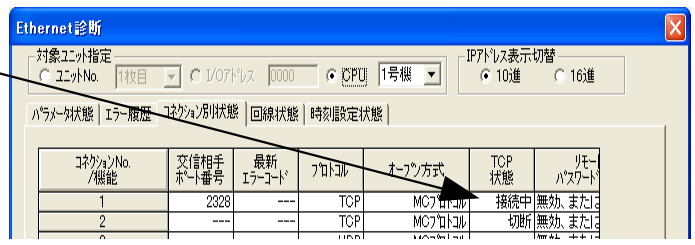
#PR TCP(0)
1

約40秒後

(e)「切断」になる前にMPCの電源をONしても通信できません。

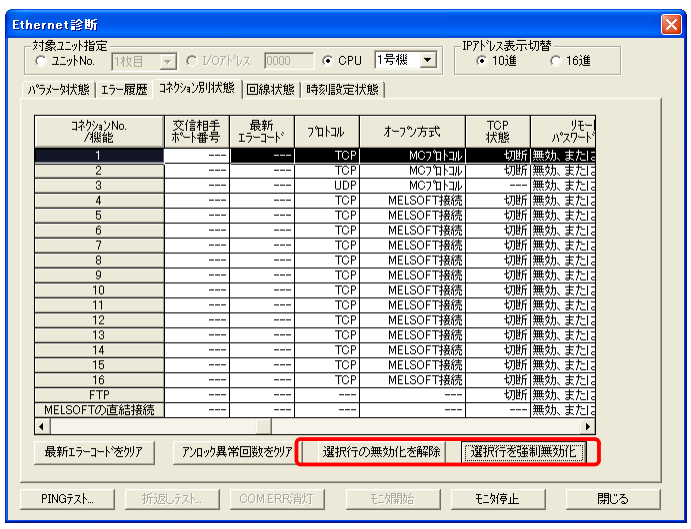
#PR TCP(0)
0

(f)そのままでも約50秒経つと「切断」になり(※)、MPC電源OFF/ONで「接続中」になります。



※「切断」にならないこともあるようです。その場合はMPC電源を40秒以上OFFします。

(g)GX DeveloperでClose,Openする方法。
上記(e)の場合でも「選択行を強制無効化」で「切断」となり、「選択行の無効化を解除」で「接続中」となります。
(これをラダーで出来るか不明)



これは生存確認の機能です(おそらく(4))。
(QnUCPUユーザーズマニュアル(内蔵Ethernetポート通信編)より引用)

(2) KeepAlive による確認

プロトコルを TCP に設定した場合は，KeepAlive による確認を行います。(KeepAlive 用 ACK 伝文に対する応答）相手機器からの最後の伝文受信から，5 秒後に生存確認用伝文を送り，相手機器からの応答有無を確認します。応答がないときは，さらに 5 秒間隔で生存確認用伝文を送ります。45 秒間応答が確認できない場合，相手機器が生存していないものとしてコネクションを切断します。相手機器が TCP KeepAlive 機能に対応していない場合は，コネクションが切断されることがあります。

(3) 設定を超える接続

パラメータの Ethernet 構成設定またはオープン設定での設定数を超える接続は行わないでください。設定数を超える TCP 接続をパソコンから行った場合，アプリケーションによっては下記の状態になることがあります。

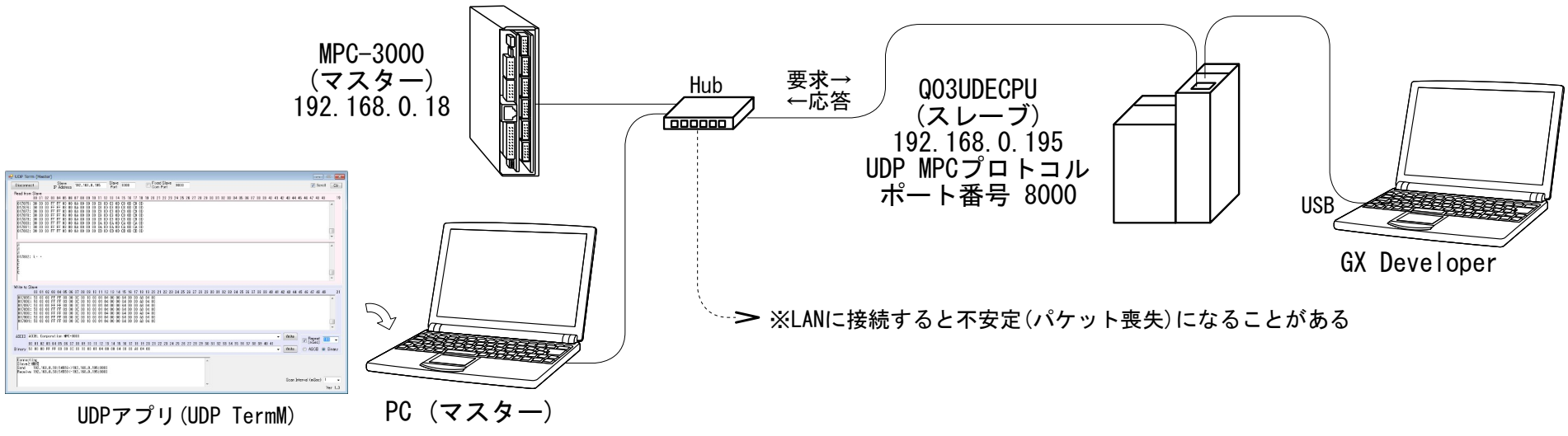
- タイムアウトエラーが発生する時間が延びる
- 通信中のいずれかの相手機器で，突然タイムアウトエラーが発生する

(4) TCP コネクションの再送処理

TCP コネクションでは，送信に対して相手機器から TCP プロトコルの ACK 応答がない場合，再送処理を行います。0.3 秒から開始して，0.6 秒，1.2 秒，2.4 秒，4.8 秒，9.6 秒後の 6 回の再送を行います。最後の再送の後，19.2 秒の間で TCP プロトコルの ACK 応答がない場合は，相手機器異常としてコネクションを切断します。(合計すると 38.1 秒で相手機器異常としてコネクションを切断します。)

■ 同一ポートUDP通信例

- （こんなことして良いのかな?）Q03の同一UDPポートに、MPCとパソコンからアクセスします。
- サンプル作成: MPC-3000(SH2A) BL/I 2.00_12 2018/12/19



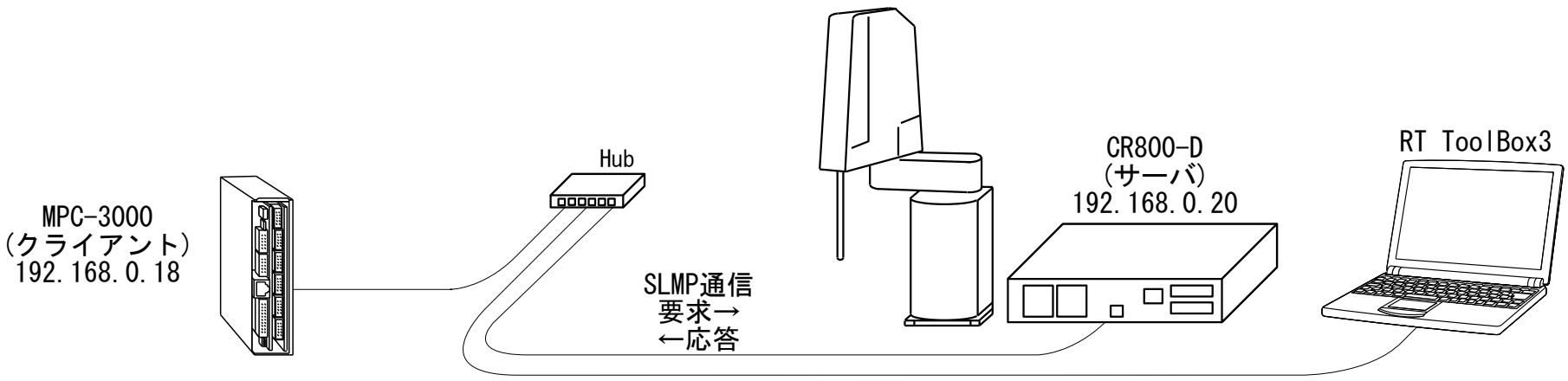
GX Developer

UDP
MCプロトコル
1F40H(8000)

- MPC-3000プログラム例

```
SET_IP 192 168 0 18
SET_IP IP_MASK 255 255 255 0
SET_IP IP_GATE 0 0 0 0
SET_IP UDP PACKET 8000

IP_CONV 192 168 0 195 plc                  /* Q03のIPアドレスを変数plcに代入
SET_DEST plc UDP_XO$                       /* 送信先と送信に使う文字列変数を紐付け

delay=100
leng=100
count=0
DO
  count=count+1
  FILL X(100) 100 count
  Q3E UDP|0 WRITE_BULK "D" 100 X(100) leng  /* X(100)~leng個 ->D100~leng個 サイズはワード(2byte)
  TIME delay
  Q3E UDP|0 READ_BULK "D" 100 Y(100) leng   /* D100~leng個->Y(100)~leng個 サイズはワード(2byte)
  TIME delay
  IP_CONV IPA(UDP_RO$)                     /* 送信元のIPアドレス表示
  FOR i=100 TO 99+leng
    IF X(j)<>Y(i) THEN
      PR "Mismatch"
      END
    END_IF
  NEXT
LOOP
```

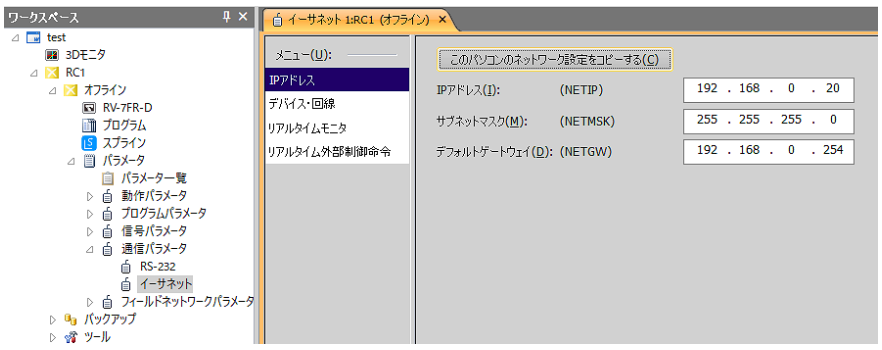
三菱ロボットコントローラ CR800-Dとの通信



- RT ToolBox3 設定例

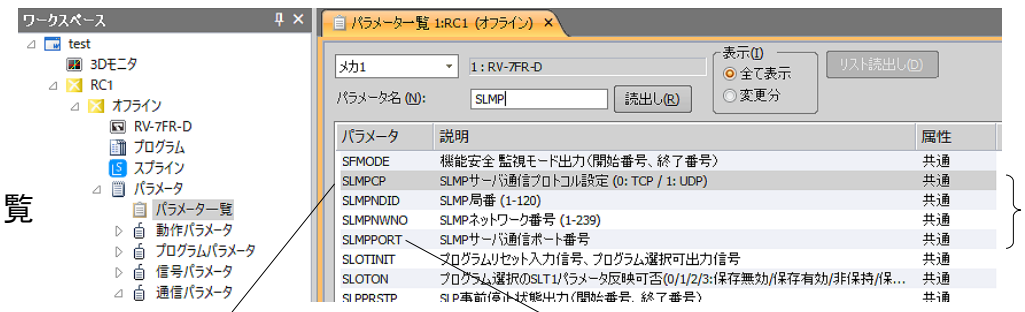
※ 設定変更、MPC電源OFFや通信切断をしたらCR800を再起動

イーサネット



自己IPアドレス等の設定

パラメーター覧



SLMPパラメータの設定

SLMPCP 通信プロトコル
0:TCP

SLMPPORT 通信ポート番号
45237(初期値)

(SLMPNDID,SLMPNWNOは初期値のまま)

※ CR800-D はロボットコントローラー側のプログラムでこのメモリエリアを読み書きできません。
CR800-R、CR800-Qはできるようです。

- MPC-3000プログラム例

```
SET_IP 192 168 0 18
SET_IP IP_MASK 255 255 255 0
SET_IP IP_GATE 0 0 0 0
SET_IP TCP PACKET 45237 192 168 0 20      /* CR800-Dがスレーブ

j=0
DO
  j=j+1
  FILL X(100) 4 j 1
  Q3E TCP|0 WRITE_BULK "D" 100 X(100) 4   /* X(100)~X(103) -> CR800 D100~D103
  Q3E TCP|0 READ_BULK "D" 100 Y(100) 4    /* Y(100)~Y(103) <- CR800 D100~D103
  FOR i=100 TO 103
    PRX X(i) Y(i)
  NEXT
  TIME 500
LOOP
```

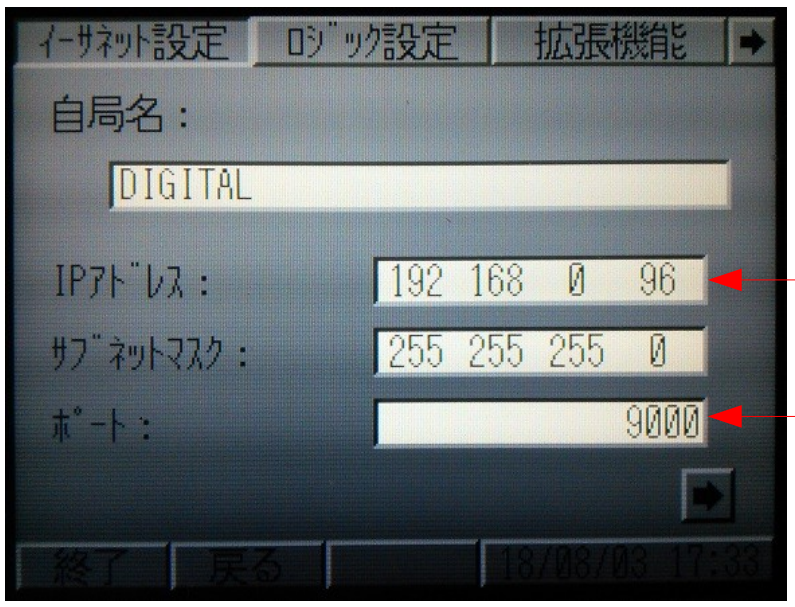
Proface GP4301 MEWTOCOL7接続の設定例

■ MPC-3000 の宣言

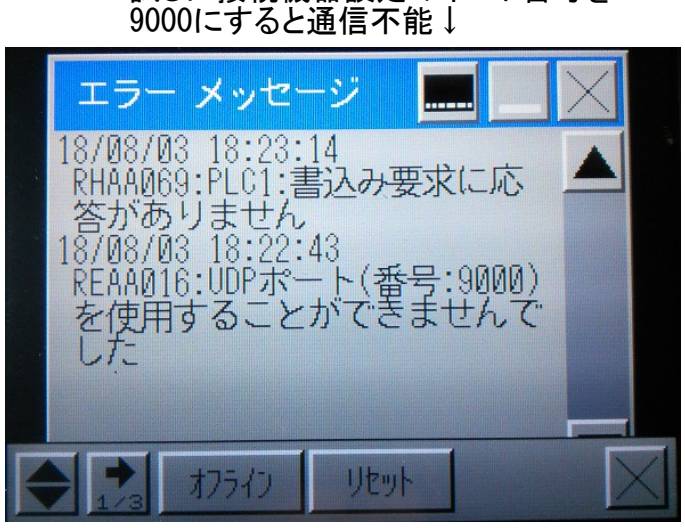
```
SET_IP 192 168 0 20 255 255 255 0 192 168 0 248
SET_IP UDP MEWTOCOL7 8000    /* MPCがGPから受信するポート番号
```

■ GP-Pro EX の設定

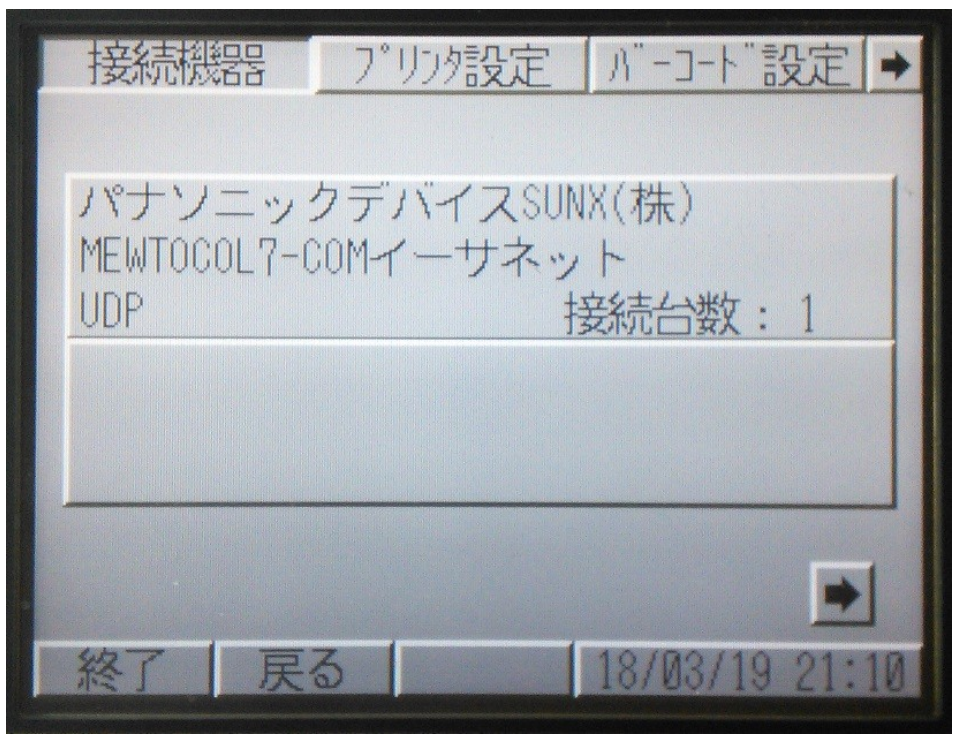
GPの オフライン ＞ 本体設定 ＞ イーサネット設定

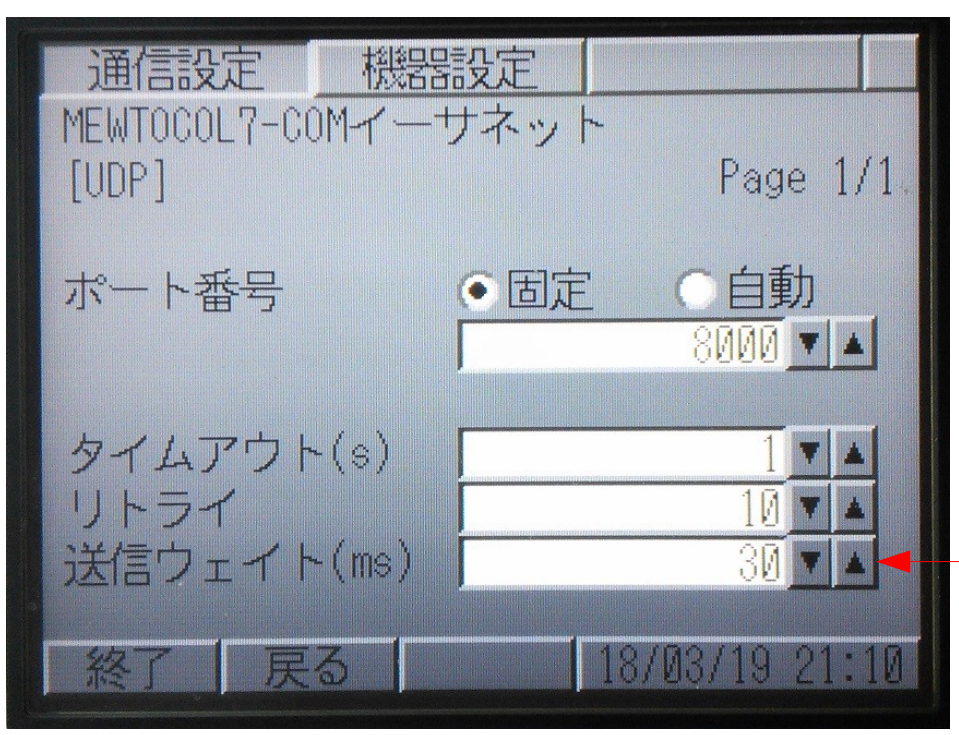


同一設定不可
試しに接続機器設定のポート番号を
9000にすると通信不能↓

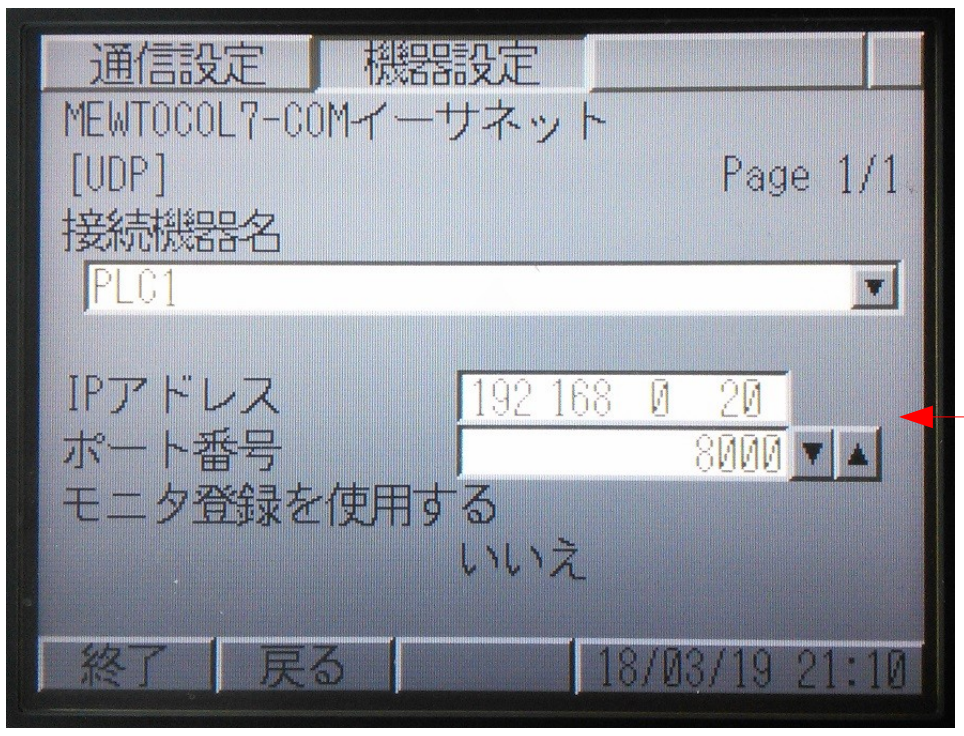


■ GPのオフライン設定





KEYENCEのタッチパネルは初期値 0
適宜調整して下さい



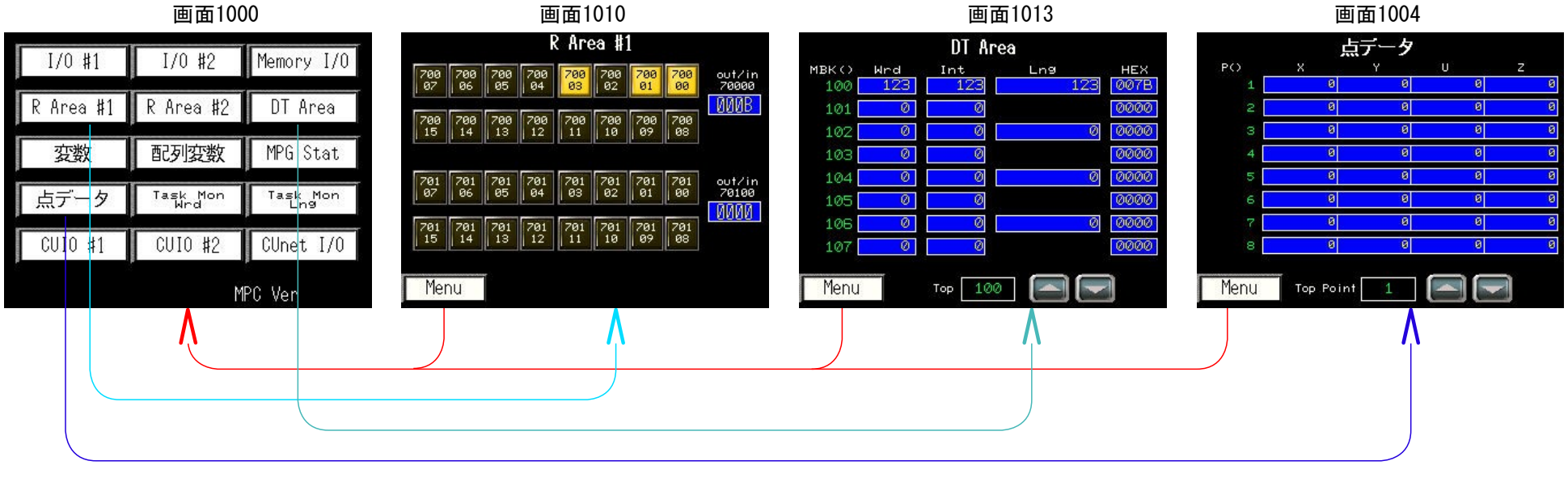
MPCのIPアドレス と ポート番号

■ 通信確立時のMPC SET_IP状態

```
#SET_IP
IPadrs = 192.168.0.20
 MASK  = 255.255.255.0
Gateway= 192.168.0.248
OFF Delay ACK
MACadrs= 001EC08DAB6E

1 : =  UDP   MEWTOCOL7     8000      Slave
```

■ Example MPC Program1

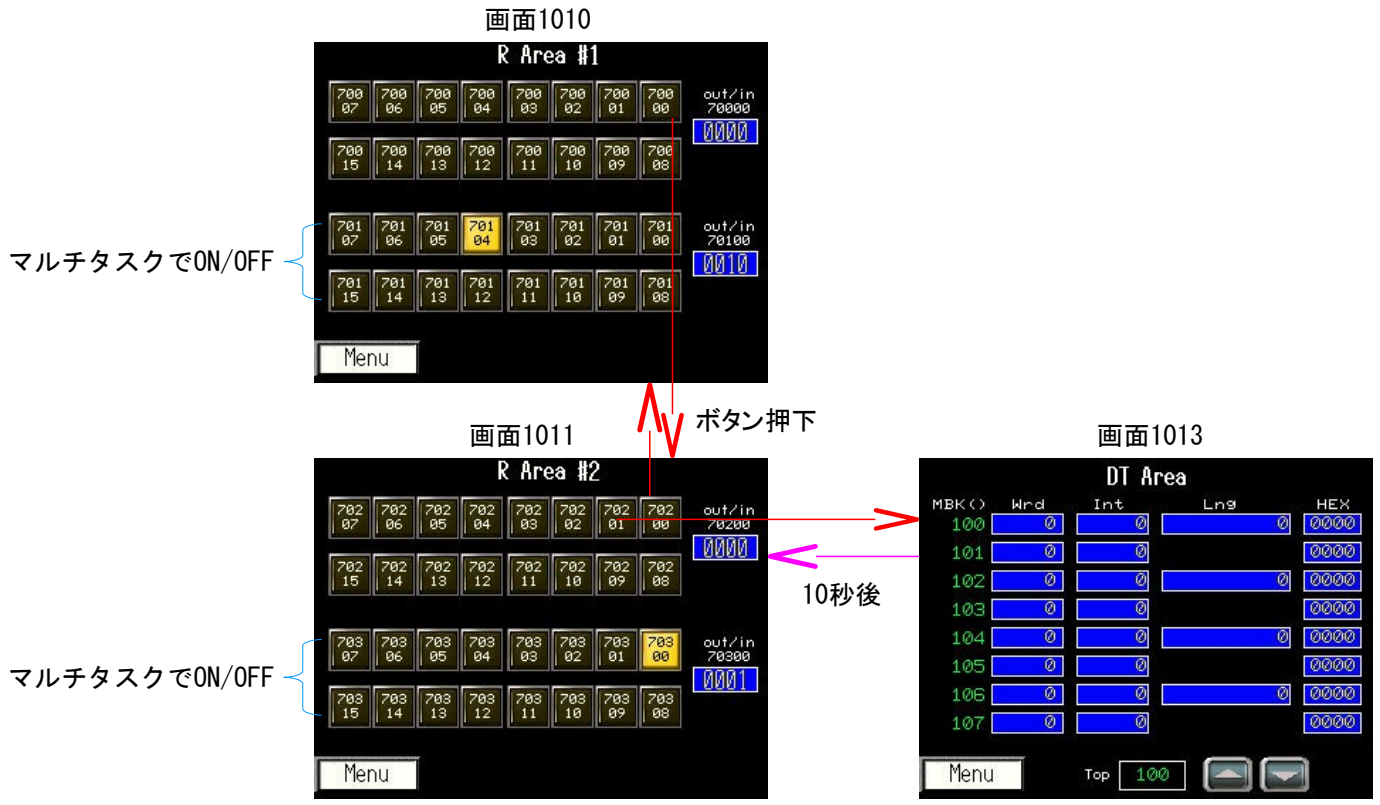
メニューで選択した画面に対応して動作します。




```
/* 下記 SW( )、MBK( )はMPCの表記、[]はGPの表記
/* Menu画面の「R Area#1」,「DT Area」または「点データ」ボタン で画面を切り替えて下さい。
/* サイズオプション ~Wrd(または省略時):符号無2byte, ~Int:符号付2byte, ~Lng:符号付4byte
SET_IP 192 168 0 20 255 255 255 0 192 168 0 248
SET_IP UDP MEWTOCOL7 8000


MBK(8)=1                /* 初期画面 MBK(8) [DT8] は画面切替
WAIT MBK(0)==1          /* MBK(0)[DT0]は現在表示中の画面番号
TIME 500
WHILE MBK(0)<>1000
  MBK(8)=1000           /* Menu画面
WEND

DO

  PRINT "Menu"
  WAIT MBK(0)<>1000     /* Menu画面から切り替わるのを待つ
  gppage=MBK(0)

  SELECT_CASE gppage

    CASE 1010
      PRINT "R Area #1"
      i=0
      WHILE MBK(0)<>1000  /* 画面下の「Menu」ボタン押下まで繰り返し
        OUT i 70000~Wrd
        /* IN(70000~Wrd)は i の値で、IN(70100~Wrd) はパネルのボタン押下で変化する
        PRINT "IN(70000~Wrd) IN(70100~Wrd)=" HEX$(IN(70000~Wrd)) HEX$(IN(70100~Wrd))
        TIME 500
        INC i
      WEND

    CASE 1013
      PRINT "DT Area"
      MBK(8056)=100       /* 標示先頭アドレス
      i=0
      WHILE MBK(0)<>1000
        MBK(100)=i
        /* MBK(100)は i の値で、MBK(102) はパネルの数値表示器入力で変化する
        PRINT "MBK(100) , MBK(102) Wrd Int Lng =" MBK(100) "," MBK(102~Wrd) MBK(102~Int) MBK(102~Lng)
        TIME 500
        INC i
      WEND

    CASE 1004
      PRINT "Point Data"
      MBK(8143)=1       /* 標示先頭アドレス
      i=0
      WHILE MBK(0)<>1000
        FOR p=1 TO 8
          /* パネルの点データの値が変化する
          SETP p i i*-1 i i*-1
        NEXT
        TIME 100
        INC i
      WEND

    CASE_ELSE
      PRINT "Page " gppage
      WAIT MBK(0)==1000
  END_SELECT

LOOP
```

■ Example MPC Program2

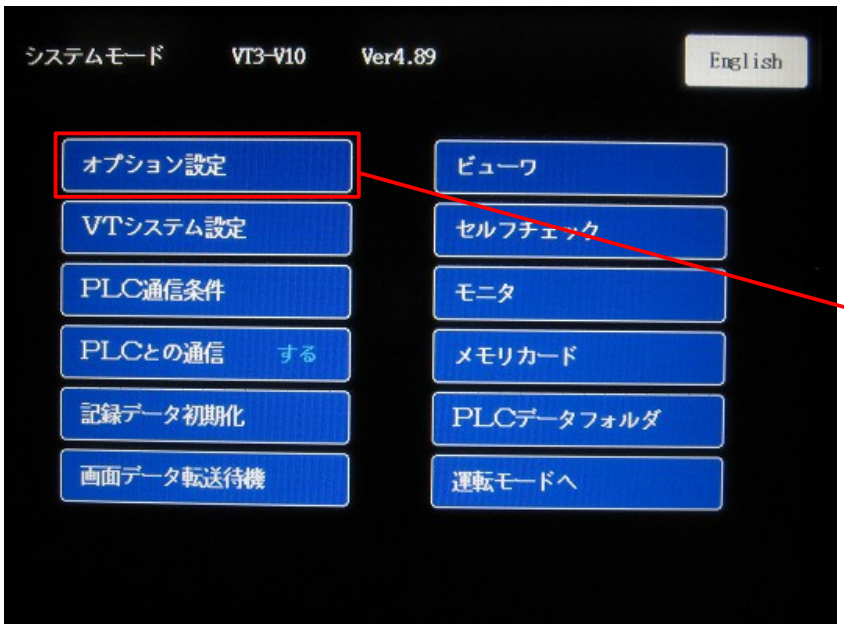
ビットスイッチ押下で画面が切替わります。画面1013から1011には自動的に戻ります。

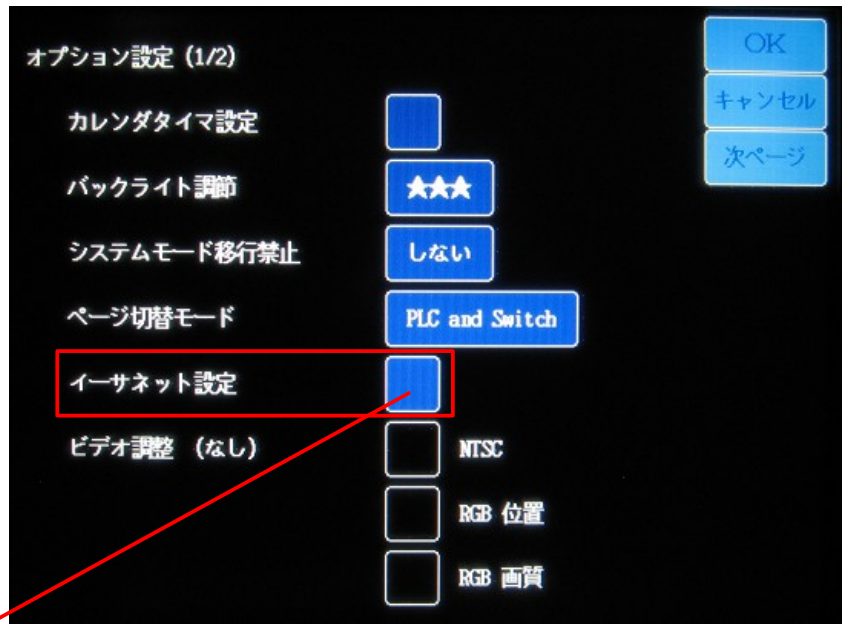


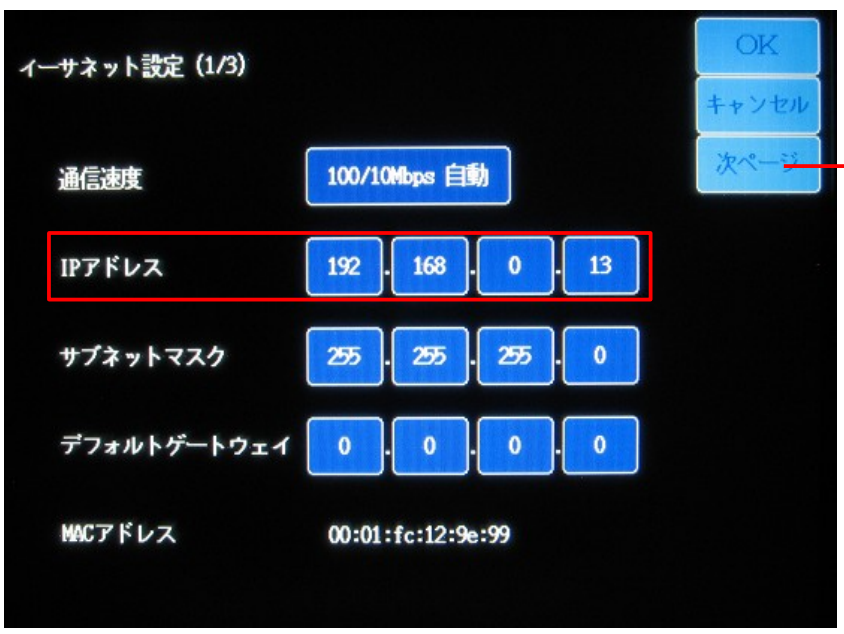

```
/* 下記 SW()、MBK()はMPCの表記、[]はGPの表記
/* 実行中、各画面の「Menu」ボタンは操作禁止

SET_IP 192 168 0 20 255 255 255 0 192 168 0 248
SET_IP UDP MEWTOCOL7 8000

MBK(8)=1                /* 初期画面 MBK(8) [DT8] は画面切替
WAIT MBK(0)==1          /* MBK(0)[DT0]は現在表示中の画面番号
TIME 500
WHILE MBK(0)<>1010
  MBK(8)=1010           /* R_Area#1画面
WEND

QUIT_FORK 1 *R_Area1
QUIT_FORK 2 *R_Area2

DO

  WAIT SW(70000)==1     /* R_Area#1 SW(70000) [R00000]
  OFF 70000
  MBK(8)=1011           /* R_Area#2画面に移行

  DO
    WAIT IN(70200)<>0   /* SW(70200)[R00020]~SW(70207)[R00027] どれかのボタン押下待ち
    SELECT_CASE VOID
      CASE SW(70200)
        OFF 70200
        MBK(8)=1010     /* R_Area#1画面に戻る
        BREAK           /* Exit DO~LOOP
      CASE SW(70201)
        OFF 70201
        MBK(8)=1013     /* DT Area画面に移行
        MBK(8056)=100   /* 標示先頭アドレス
        FOR i_=10 TO 0 STEP -1    /* Count Down
          FOR j_=100 TO 107
            MBK(j_)=i_  /* MBK(100)[DT100]~MBK(107)[DT107] に値を入れる
          NEXT
          dt=MBK(100)   /* MBK(100)[DT100]の読み込み
          PR "残り" dt "Sec"
          TIME 1000
        NEXT
        MBK(8)=1011     /* 10数えてR_Area#2画面に戻る
      CASE_ELSE
        OUT 0 70200
    END_SELECT
  LOOP

LOOP

*R_Area1

DO
  FOR i_=70100 TO 70115
    ON i_
    TIME 100
    OFF i_
  NEXT
LOOP

*R_Area2

DO
  FOR i_=0 TO &HFFFF
    OUT i_ 70300~Wrd
    TIME 100
  NEXT
LOOP
```

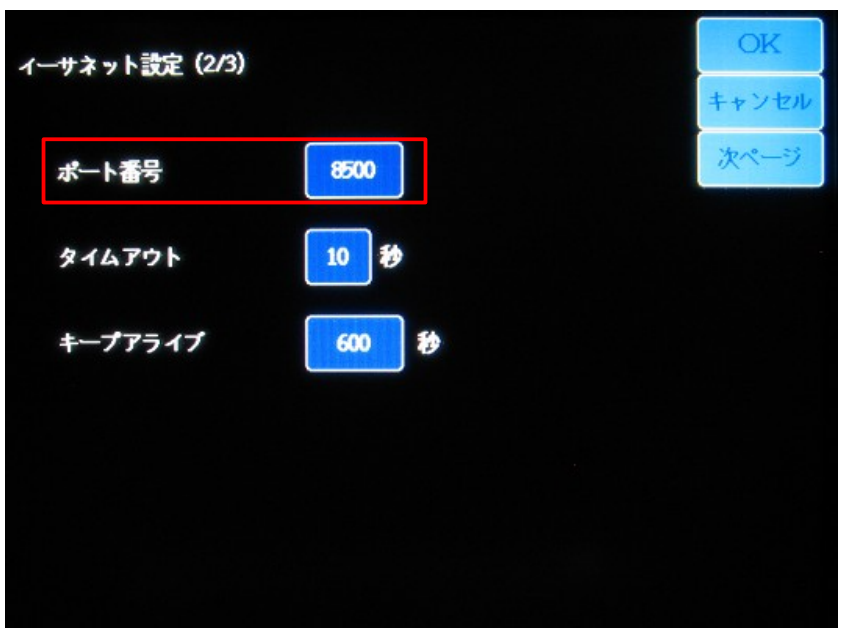
KEYENCE VT3 MEWTOCOL7接続の設定例

■ 本体のIPアドレス、ポート番号はオフラインで設定します。(VT3はVT STUDIOで設定が出来ない?。VT5は出来るようです)

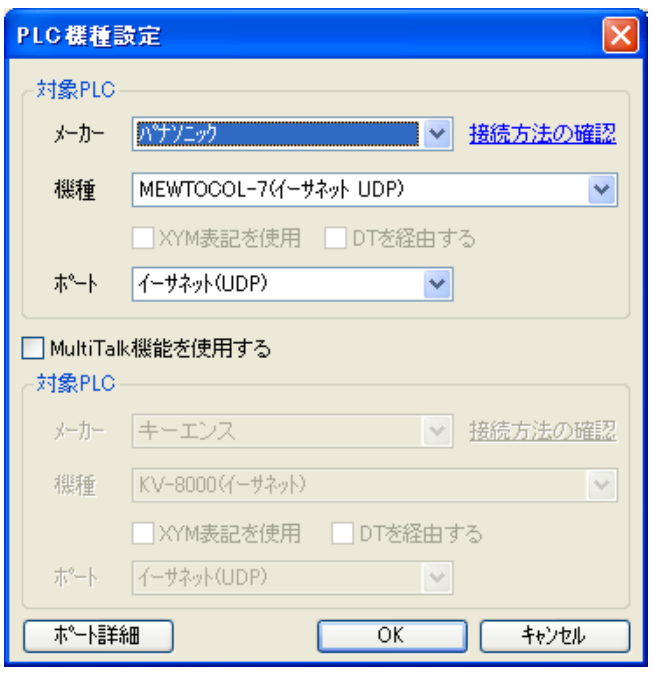


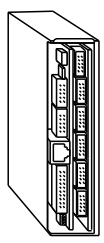






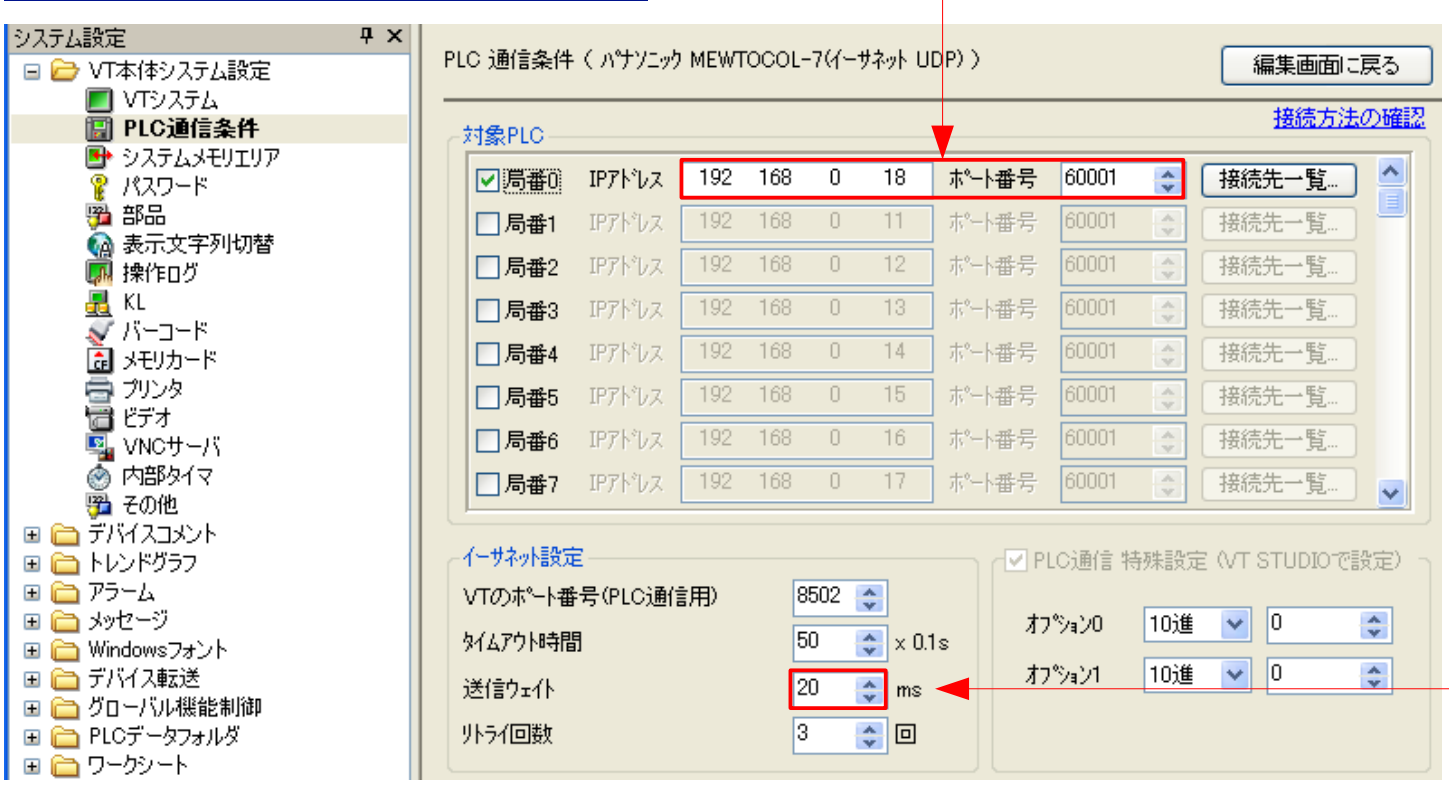
■ VT STUDIO Ver 7.10(VT3シリーズ) の設定





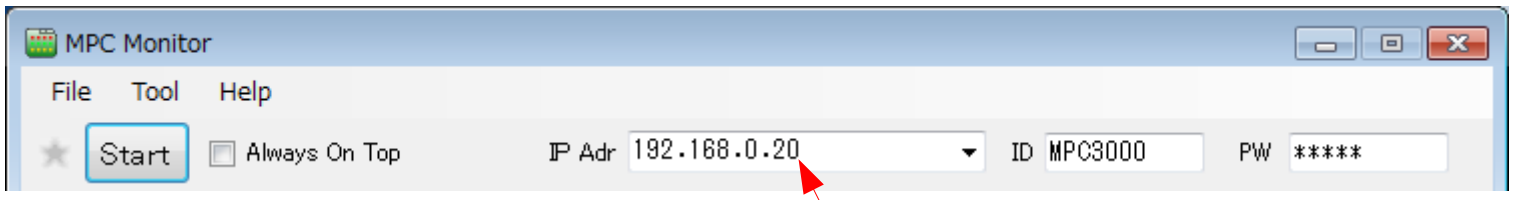
設定
SET_IP 192 168 0 18 255 255 255 0 0 0 0 0
SET_IP UDP MEWTOCOL7 60001

MPCのIP Address、Port番号



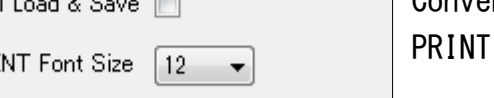
初期値は0です。
適宜調整してください。

MPC Monitor Telnetの設定



MPC-3000のIPアドレス
MPCの宣言例） SET_IP 192 168 0 20 255 255 255 0 192 168 0 248

ポート番号の指定例(省略時は23)



Connection Timeout 接続時タイムアウト
Read Timeout コマンド応答タイムアウト
Send Wait 送信ウェイト
MMC Load Timeout 高速LOADモードタイムアウト

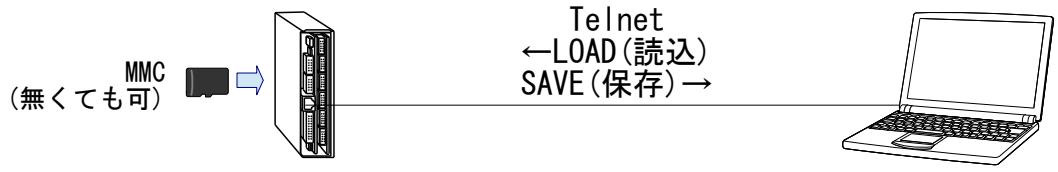
Hold Form Size フォームサイズ保持
No Version Check 接続時バージョンチェック無し
Conventional Load & Save 従来のLOADとSAVE
PRINT Font Size FILEタブ表示のフォントサイズ

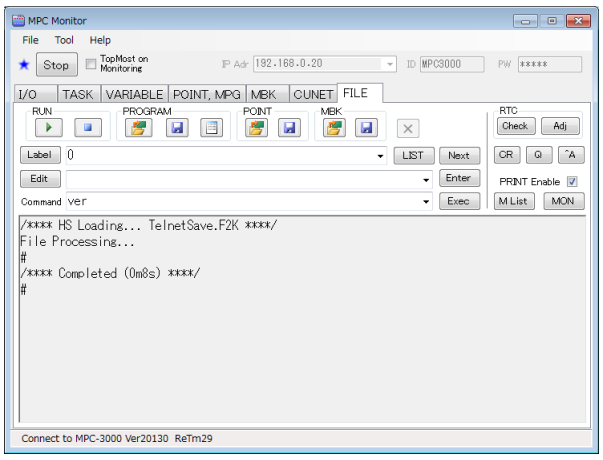
(Ver1.06)

接続中に Timeout で停止する場合は Read Timeout の時間を大きくしてみてください。

■ プログラム読込と保存(Telnet)

[LOAD] FROM(Flash ROM)またはMMC経由でプログラムを読み込みます。従来方法より時間を大幅に短縮します。
[SAVE] プログラム保存も転送内容を見直しました。





MPC Monitor Telnet Ver1.04 以降
MPCが 2.01_30 以降で自動的に
HSモードになります。

「開発ツールダウンロード」のNo055です。
https://accelmpc.co.jp/downloads/dev_uty.html
ダウンロード・展開して ¥Accelフォルダにコピーして下さい。

MPC-3000(SH2A) BL/I 2.01_30 2019/06/05 以降

- Telnetの設定
SET_IP コマンドをプログラムに記述。
変更したら RUN IP_INIT<Ent> → 電源再投入。

- 設定例
SET_IP 192 168 0 20 255 255 255 0 192 168 0 248
SET_IP TCP Telnet

- 確認
#SET_IP
IPadrs = 192.168.0.20
 MASK = 255.255.255.0
Gateway= 192.168.0.248
OFF Delay ACK baud=38400
MACadrs= 001EC08DAC7A

1 : / TCP   **Telnet**   23   Slave
#

- プログラムはFROMまたはMMCにファイル転送してからMPCに取り込みます。エラーは取り込み時に表示されます。

```
/* Load Save Test
SET_IP 192 168 0 20 255 255 255 0 192 168 0 248
SET_IP TCP Telnet

abc
FOR i=100 TO 0 STEP -1
  PR i
NEXT
```

HS表示 ▼　　転送 ファイル処理

```
/**** HS Loading... TelnetSave.F2K ****/
File Processing...
5 abc
      未定義コマンド
#
/**** Completed (0m1s) ****/
#
```

- ファイル名は予約変数MMC_LOAD$で指定します。初期化後のMMC_LOAD$は"FLASH.F2K"となっており、その場合プログラムはFROM経由でLOADされます。MMC_LOAD$が"FLASH.F2K"以外なら、そのファイル名でMMCに保存されます。(FROMよりLOAD時間が長くなります)

例) MMCが挿入されており、MMC_LOAD$="ABC.F2K"としてLOADすると、MMCにABC.F2Kとして保存されます。

```
#MPCINIT
#ERASE
*
#PR MMC_LOAD$
 FLASH.F2K                                    ←ファイル名初期値
#MMC_LOAD$="ABC.F2K"                          ←ファイル名変更
/**** HS Loading... TelnetSave.F2K ****/ (0m20s)  ← LOAD
File Processing...
#
/**** Completed (0m25s) ****/
#FILES                                        ←MMC内ファイル一覧表示
MMC_Card inserted Mounted

    SYSTEM~1      2019/ 3/26 10:16 0
     ABC.F2K      2019/ 6/ 5 16:23 340790     ←保存されたプログラム
#
```

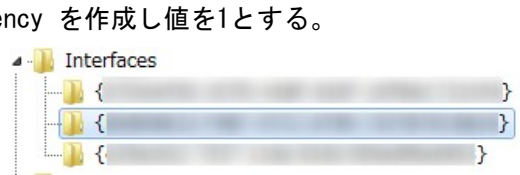
- LOAD、SAVE時間実測値 (約333kbyte 11600行のプログラム。ネットワーク環境等で変ります。)

		Win7 Desk Top (Local)	Win10 Desk Top (Local)	Win10 Note (WiFi)
LOAD	新(HS)	9s	10s	10s
	従来	2m55s	2m56s	3m01s
SAVE	新	1m08s	18s	19s
	従来	1m22s	25s	21s

- <<参考>>SAVEを更に速くするパソコンの設定　※レジストリを操作します。自己責任で行ってください。

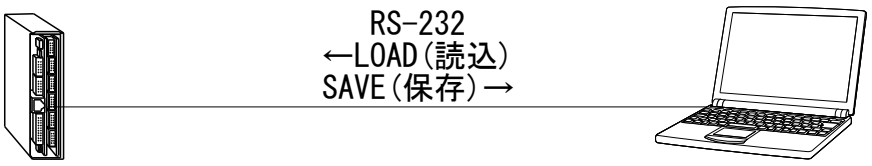
1)レジストリエディタで HKEY_LOCAL_MACHINE¥SYSTEM¥ControlSet001¥services¥Tcpip¥Parameters¥Interfaces の下から自分のIPアドレスが記載されているキーを探す。(ControlSetxxxが複数有る場合は HKEY_LOCAL_MACHINE¥SYSTEM¥Select の Current の値を参照する。)
2)逆クリック→新規→DWORD値で名前 TcpAckFrequency を作成し値を1とする。
3)PCを再起動する。

結果)上記のSAVE時間が4sになった。



■ プログラム読込と保存(RS-232)

通信レート115200bpsに対応しました。



MPC-3000(SH2A) BL/I 2.01_30 2019/05/29 以降

- Baud Rateの設定

SET_IP コマンドをプログラムに記述。
変更したら RUN IP_INIT<Ent> → 電源再投入。

- Baud Rate設定

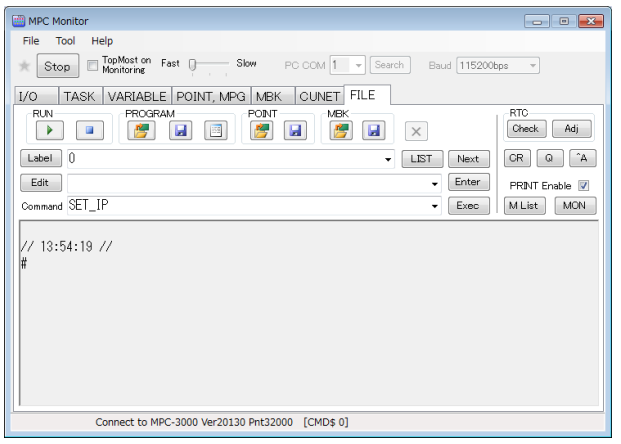
SET_IP 115200

- 確認

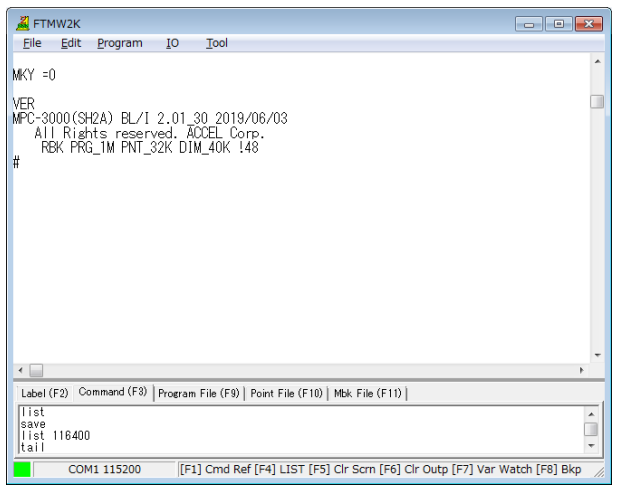
```
#SET_IP
IPadrs = 192.168.0.20
 MASK  = 255.255.255.0
Gateway= 192.168.0.248
OFF Delay ACK  baud=115200
MACadrs= 001EC08DAC7A
```

- 38400に戻すには

```
SET_IP 38400
```



MPC Monitor

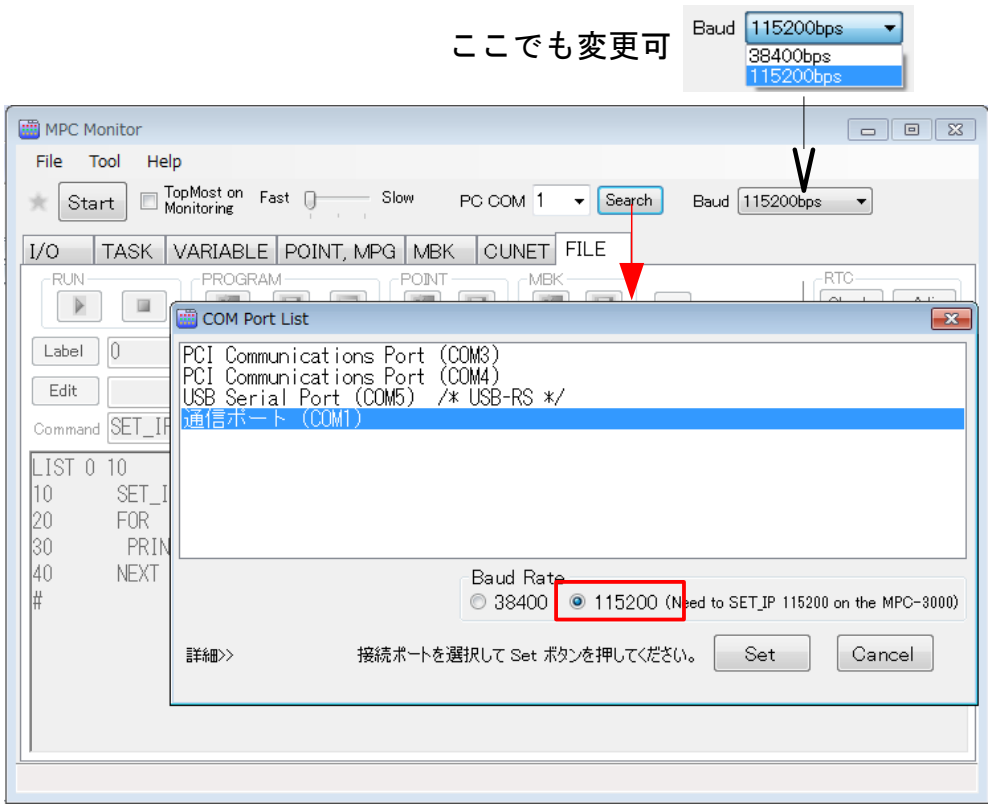


FTMW2K

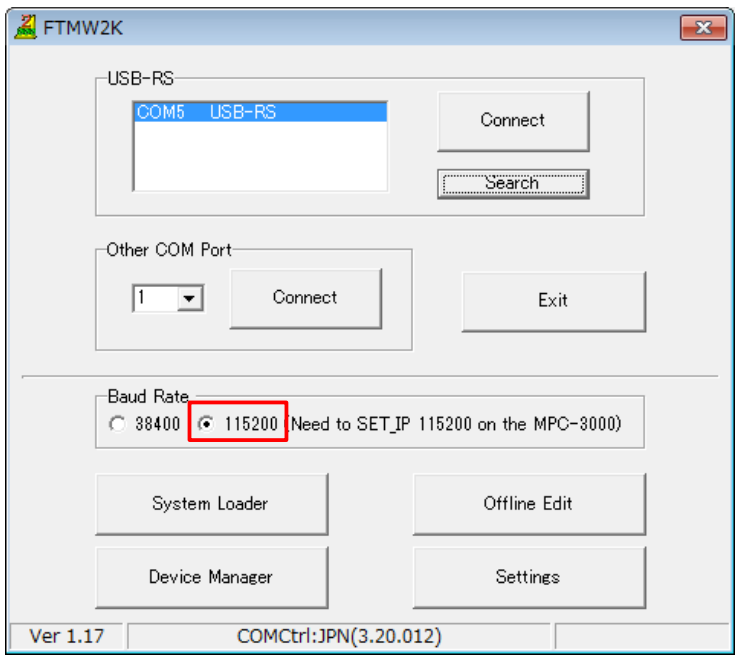
「開発ツールダウンロード」 のNo050、No040です。
https://accelmpc.co.jp/downloads/dev_uty.html
ダウンロード・展開して ¥Accelフォルダにコピーして下さい。

- MPC Monitor、FTMW2K の設定。(自動設定ではありません)

ここでも変更可



MPC Monitor (RS-232) Ver1.75 以降

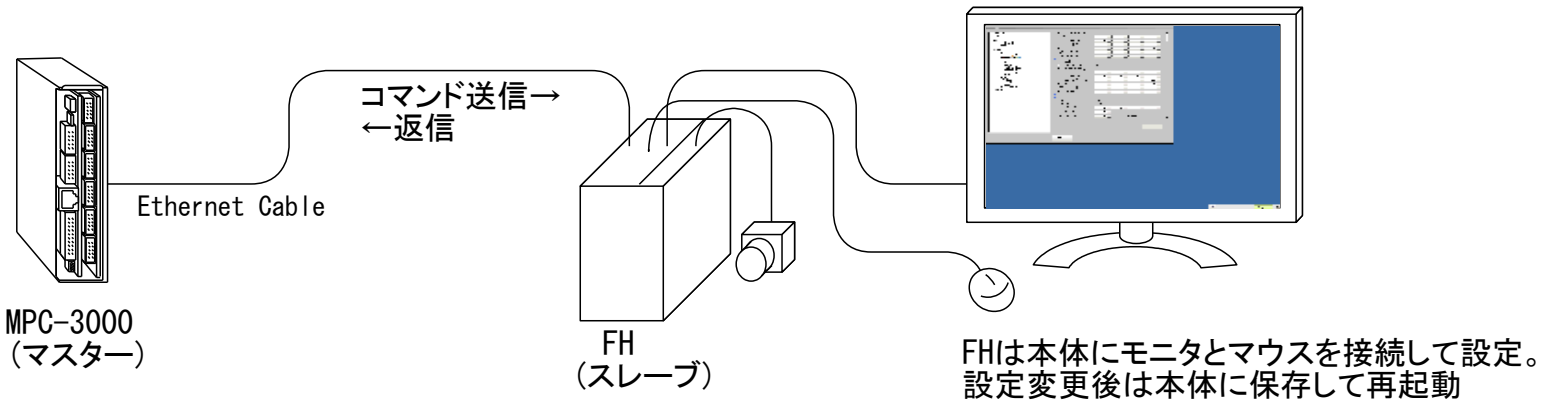


FTMW2K Ver1.17 以降

- LOAD、SAVE時間実測値 (MPC Monitor 約333kbyte 11600行のプログラム)

	bps	所要時間
LOAD	115200	50s
	38400	1m53s
SAVE	115200	40s
	38400	1m51s

OMRON 画像処理システム FHシリーズ



■ TCP通信

- MPCの宣言

```
SET_IP 192 168 0 19 255 255 255 0 192 168 0 248
SET_IP TCP PACKET 9000 192 168 0 1
```

- SET_IPコマンドで確認

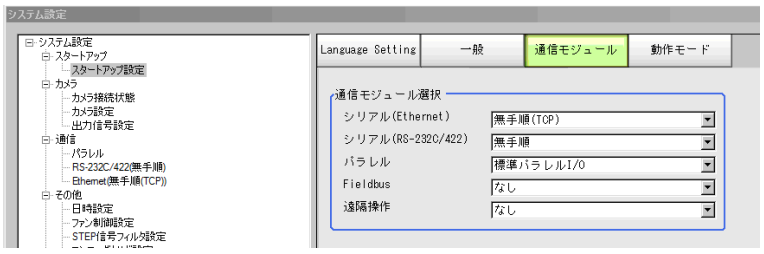
```
1 : =  TCP   PACKET  9000 <TCP_XO$><TCP_RO$> Master 192.168.0.1 ref_time = 50
```

- 送受信

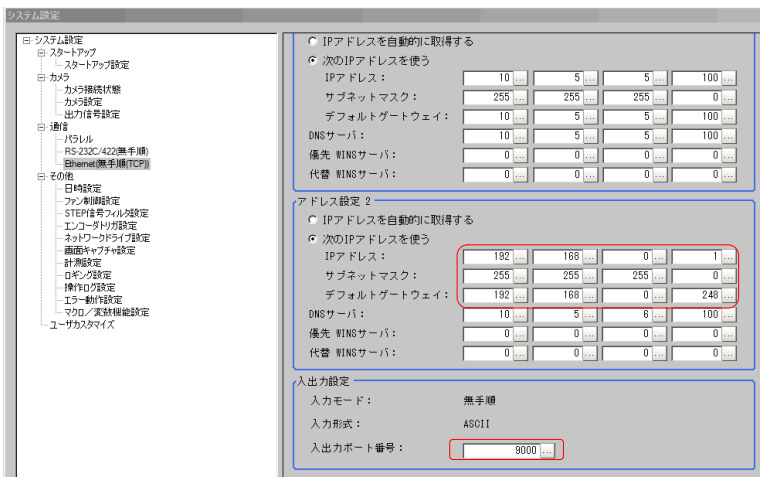
```
#TCP_RO$=""     /* 受信文字列クリア
#TCP_XO$="M"    /* 送信(デリミタ不要)
#st$=TCP_RO$    /* 受信
#PR st$
 OK             /* OK<CR> と返ってくる
#
```

※ OK応答の後にデータが来る場合は
「連続パケット受信と文字列分割例」を参照してください。

- FHの設定



無手順(TCP)



FH本体の設定

MPCと通信するポート

■ UDP通信

- プログラム例　(BL/I 2.00_09 2018/11/07以降)

```
SET_IP 192 168 0 20 255 255 255 0 192 168 0 248 /* 自己IP設定
SET_IP UDP PACKET 9000                  /* UDP Port番号
IP_CONV 192 168 0 1 OmronFh             /* 変数 OmronFh に FH のIP 192.168.0.1 を入れる
SET_DEST OmronFh UDP_XO$                /* 宛先設定
DO
  UDP_RO$=""                            /* 受信文字列クリア
  UDP_XO$="M"                           /* ASCII文字列送信
  timer_=10
  WAIT (IPC(UDP_RO$)!=0)|(timer_==0)    /* 応答 or タイムアウト
  IF timer ==0 THEN
    PR "TIME OUT"
  ELSE
    receiveIP=IPA(UDP_RO$)&&hFF         /* IP最下位取得
    PR "Receive From" receiveIP
    ptr_=UDP_RO$
    rcv$=PTR$(IPC(UDP_RO$))
    PR rcv$
  END_IF
LOOP
```

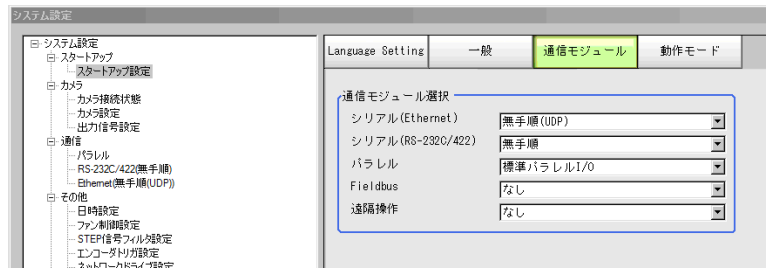
- 実行結果

```
RUN

# Receive From 1
 OK
```

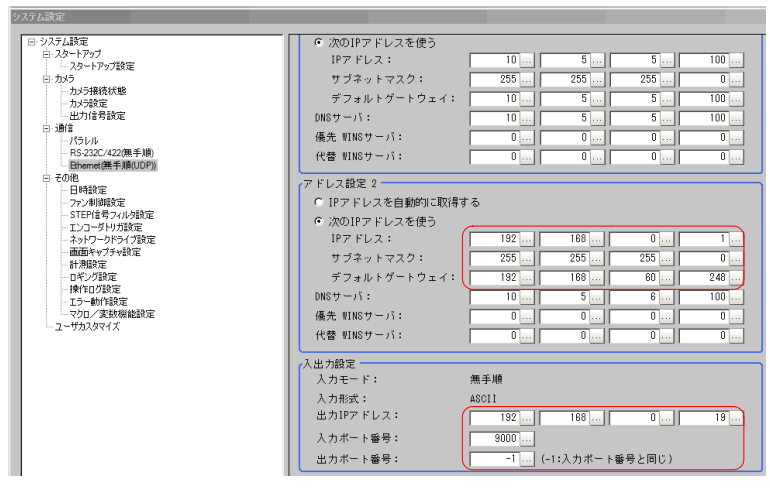
- SET_IPコマンドで設定確認

```
1 : /  UDP   PACKET        9000     <UDP_XO$><UDP_RO$>
  (↑ UDPは接続が成立しても = になりません)
```

- FHの設定



無手順(UDP)



FH本体の設定

MPCのIPアドレス
と通信するポート

■ ¥マーク、″(ダブルコーテーション)を含む文字列の送信方法

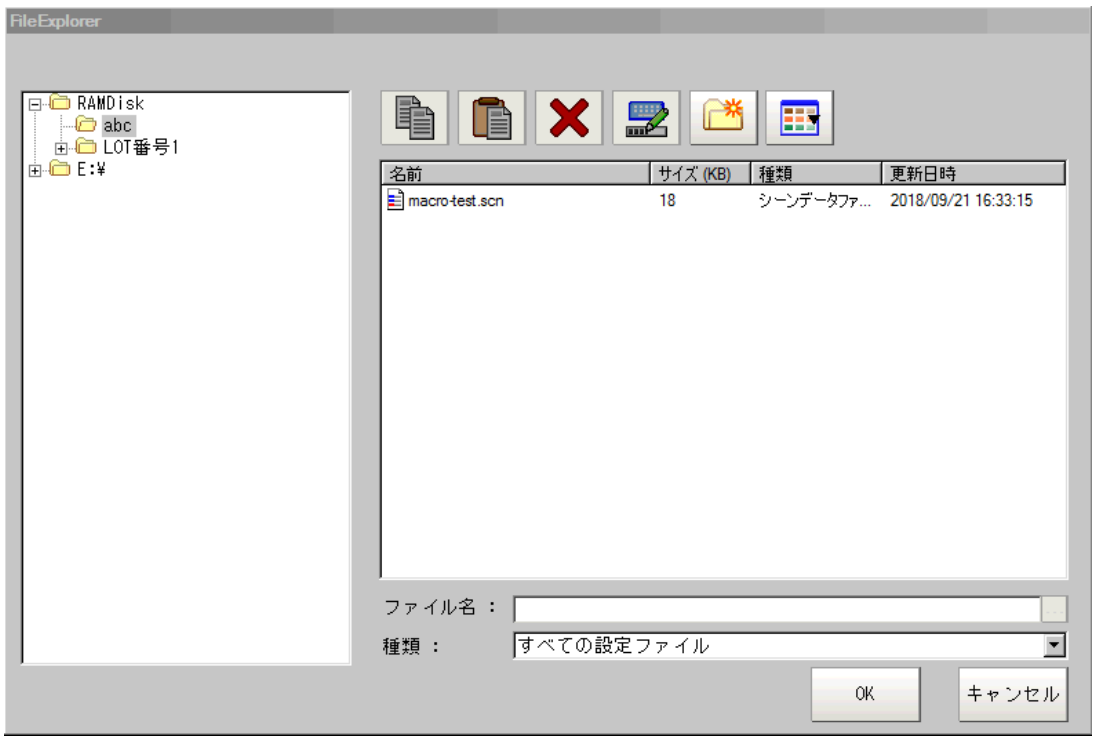
- FHのコマンドには ¥ や ″ を用いるものがあります。下記はそれらを含む文字列を出力するプログラム例です。

```
SET_IP 192 168 0 19 255 255 255 0 192 168 0 248
SET_IP TCP PACKET 9000 192 168 0 1

TCP_RO$=""          /* 受信文字列変数クリア

TCP_XO$="SCNSAVE 0 ¥x22¥¥Data¥¥RAMDisk¥¥abc¥¥macro-test.scn¥x22" /* ¥マーク出力は ¥¥、 ″(ダブルコーテーション)は¥x22

WAIT TCP_RO$<>""    /* 受信待ち

st$=TCP_RO$
PR st$
```

- スレーブに送信された文字列

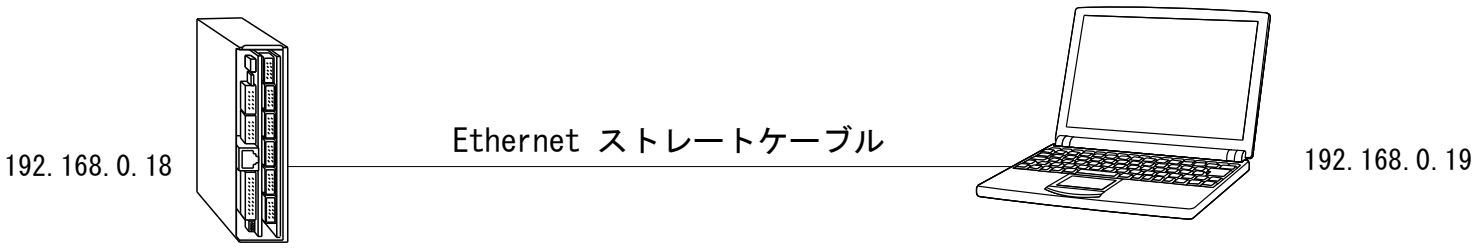
```
SCNSAVE 0 "¥Data¥RAMDisk¥abc¥macro-test.scn"
```

SCNSAVE は↓シーンファイルを保存するコマンド



MPCとパソコンの直結

- MPCとパソコンを直結してアプリケーション(VB.net)と連続通信をさせています。
- 正常に接続を確立できないPCも有ります。その場合はハブを介してみてください。
- サンプル作成: MPC-3000(SH2A) BL/I 2.00_11 2018/11/26



■ TCP(n)の状態

電源投入後
#PR TCP(0) TCP(1)
 0 0

アプリと通信中
#PR TCP(0) TCP(1)
 1 1

アプリ側切断
#PR TCP(0) TCP(1)
 0 0

MPC側切断
#dummy=TCP(-1,0)
#dummy=TCP(-1,1)
#PR TCP(0) TCP(1)
 2 2

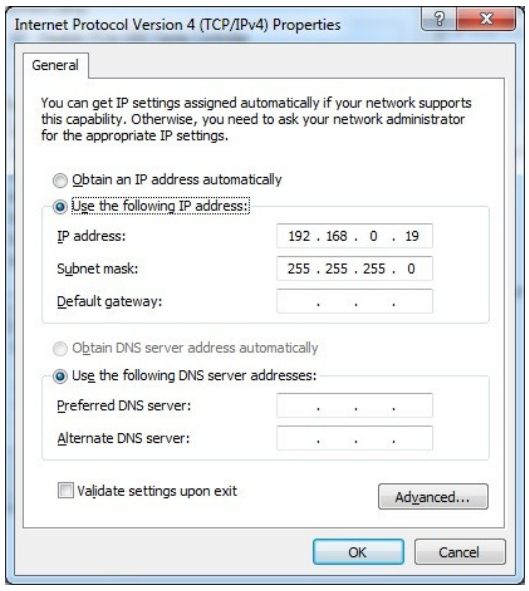
MPC側再接続
#dummy=TCP(-2,0)
#dummy=TCP(-2,1)
#PR TCP(0) TCP(1)
 0 0

アプリ側再接続
#PR TCP(0) TCP(1)
 1 1

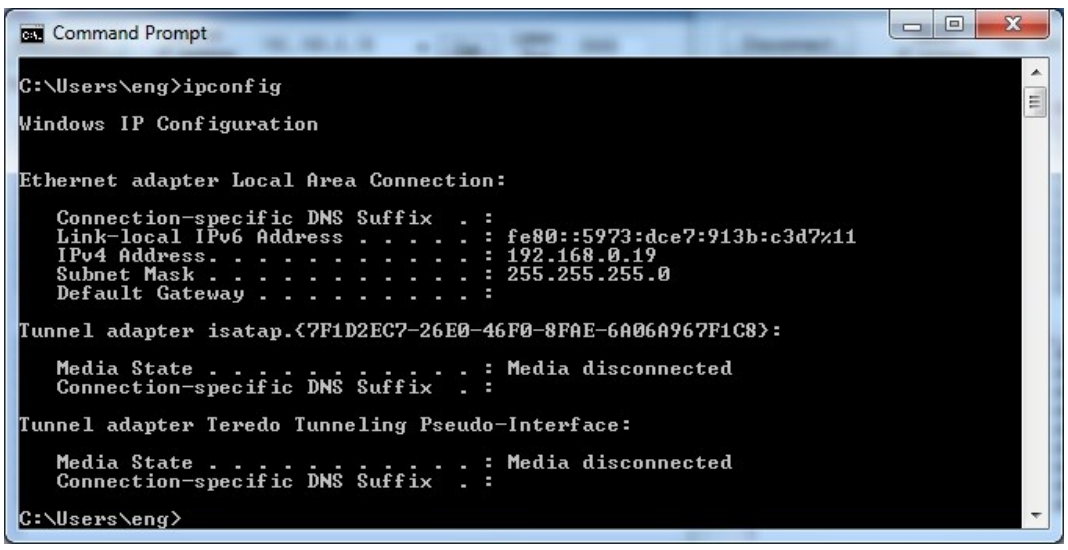
■ 動作確認MPCプログラム

```
SET_IP 192 168 0 18
SET_IP IP_MASK 255 255 255 0
SET_IP IP_GATE 0 0 0 0                /* Default Gateway設定無し
SET_IP TCP PACKET 9000 192 168 0 19 /* MPCがMaster
SET_IP TCP PACKET 10000               /* MPCがSlave
QUIT_FORK 1 *TCP_Slave

/* Master
toc=0
DO
  TCP_R0$=""                                    /* 受信文字列クリア
  master_snd$="TODAY "+DATE$(1)+" NOW "+TIME$(1)
  TCP_X0$=master_snd$                           /* Slaveアプリへ送信
  timer_=20
  WAIT (TCP_R0$<>"")|(timer_==0) /* Slaveアプリから受信 or タイムアウト
  IF timer_==0 THEN
    PR "Timeout" toc
    toc=toc+1
  END_IF
  master_rcv$=TCP_R0$                           /* 受信文字列取り込み
  IF master_snd$<>master_rcv$ THEN
    PRINT "mismatch"
    END
  END_IF
LOOP

/* Slave
*TCP_Slave
DO
  TCP_R1$=""                    /* 受信文字列クリア
  WAIT TCP_R1$<>""              /* Masterアプリから受信待ち
  slave_st$=TCP_R1$             /* 受信文字列取り込み
  TCP_X1$=slave_st$             /* Masterアプリへ送信
LOOP
```

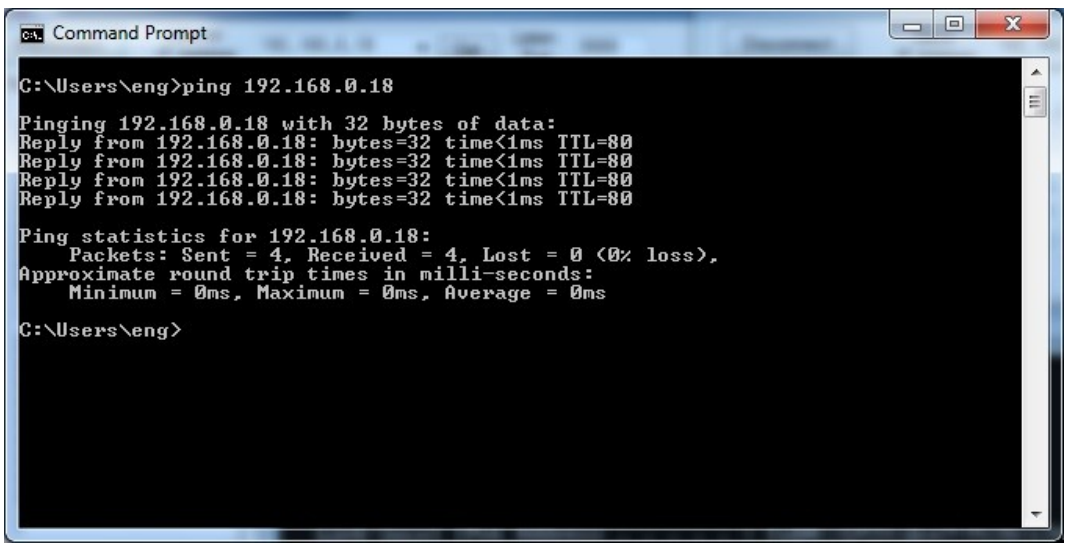
Local Area Connection



IPCONFIGでPCの設定状況確認

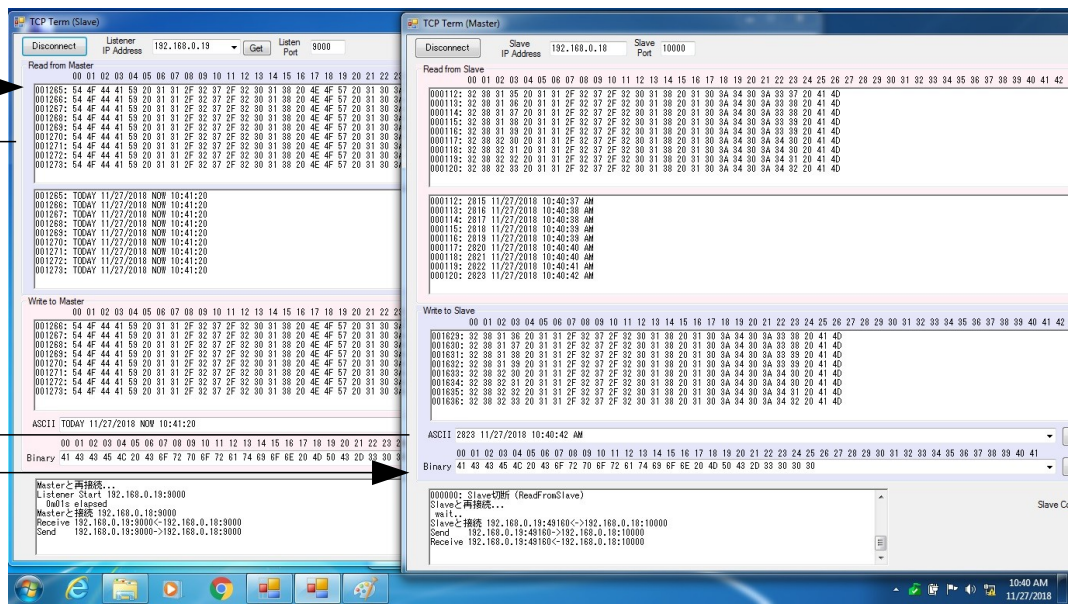


PINGでMPCとの接続確認



パソコンはSlaveアプリとMasterアプリを同時実行

TcpTermS.exe　　TcpTermM.exe



TcpTermSは受信した文字列を
MPCへループバックする

TcpTermMはMPCへ文字列を送信
MPCはそれをループバックする

■ 接続時の SET_IP情報

```
#SET_IP
IPadrs = 192.168.0.18
 MASK  = 255.255.255.0
Gateway= 0.0.0.0
OFF Delay ACK
MACadrs= 001EC08D567A

1 : =  TCP PACKET   9000  <TCP_X0$><TCP_R0$> Master 192.168.0.19 ref_time = 50
2 : =  TCP PACKET   10000 <TCP_X1$><TCP_R1$> Slave  connected-> 192.168.0.19
#
```

PACKET通信に用いる文字列変数名。
プログラムの宣言順。
$の前の数字がチャンネル番号。
TCP(n)のnもこの番号。

SET_IP 設定の更新方法

- プログラムでSET_IPの設定を変更したら下記の処理をして下さい。
- SET_IP宣言の内容が変らなければこの処理は不要です。
- RUN IP_INIT を実行するとプログラムが走ります。装置の動作にご注意下さい。
- 表示される内容は設定やバージョンにより異なります。
- サンプル作成:MPC-3000(SH2A) BL/I 2.00_03 2018/10/02

■Before

- プログラム

```
#LIST 0 10
10      SET_IP   192 168 0 18 255 255 255 0 192 168 0 248
20      SET_IP   TCP PACKET 9000 192 168 0 18
```

- SET_IPコマンドで現状確認

```
#SET_IP
IPadrs = 192.168.0.18
 MASK  = 255.255.255.0
Gateway= 192.168.0.248
OFF Delay ACK
MACadrs= 001EC08D567A

1 : /  TCP   PACKET        9000     <TCP_X0$><TCP_R0$> Master 192.168.0.18 ref_time = 50
#
```

■ After

- 異なる設定のプログラムを読み込む

```
#LIST 0 10
10      SET_IP   192 168 0 20 255 255 255 0 192 168 0 248
20      SET_IP   UDP PACKET 9000 192 168 0 67
```

・RUN IP_INIT を実行する

```
#RUN IP_INIT
Eraes IP settings
Refresh IP settings

Erase Flash
Compiling
  0 Labels
-Pass_1 completed
-Pass_2 completed
-Pass_3 completed
-GetPrgSum 37366027
-----------------
All the tasks have been quit !!
Wake TCP
Wake UDP
 ID:01 :   TASK 46 :  UDP PACKET <UDP_X0$><UDP_R0$>
Wake Serial
#
```

- MPCの電源を入れ直す

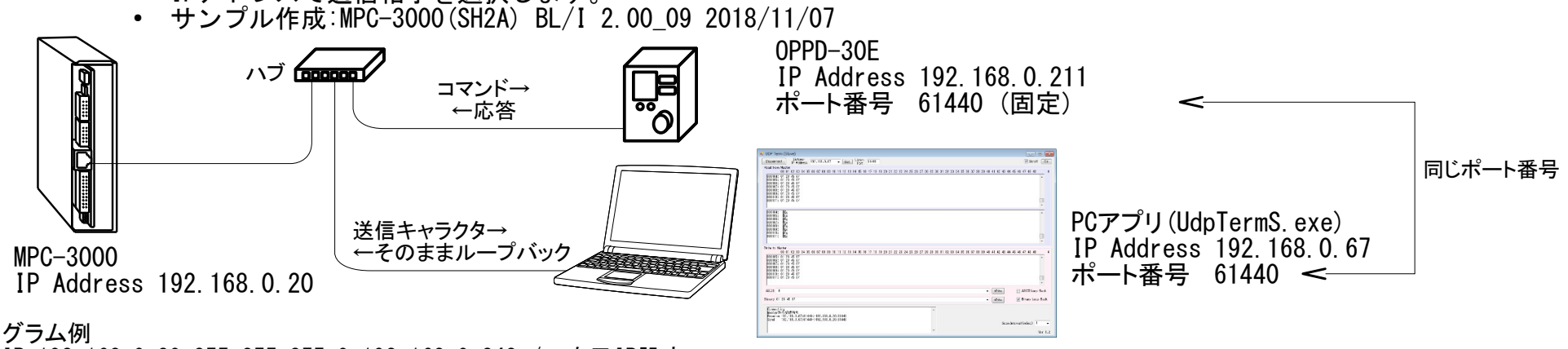
```
MKY =1
+++++++++++++++++
Detect = 24AA02E48
IPadrs = 192.168.0.20
 MASK  = 255.255.255.0
Gateway= 192.168.0.248
OFF Delay ACK
MACadrs= 001EC08D567A
Ether Opened
TCP_STACK_RAM=11768
TCP Init Completed
Wake TCP
Wake UDP
 ID:01 :   TASK 46 :  UDP PACKET <UDP_X0$><UDP_R0$>
Wake Serial
USB_I/F  Ver1.12
iMPC-3000
#
```

- 状態を確認

```
#SET_IP
IPadrs = 192.168.0.20
 MASK  = 255.255.255.0
Gateway= 192.168.0.248
OFF Delay ACK
MACadrs= 001EC08D567A

1 : /  UDP   PACKET        9000     <UDP_X0$><UDP_R0$> Master 192.168.0.67 ref_time = 50
#
```

オプテックス・エフエー(株) LED照明コントローラ OPPD とのUDPパケット通信

- OPPD-30EはUDP/IP、ポート番号は61440固定です。
- このサンプルはOPPDを2台接続することを想定していますが、1台は実機、1台はPCで代用しています。
- IPアドレスで通信相手を選択します。
- サンプル作成:MPC-3000(SH2A) BL/I 2.00_09 2018/11/07



■ プログラム例

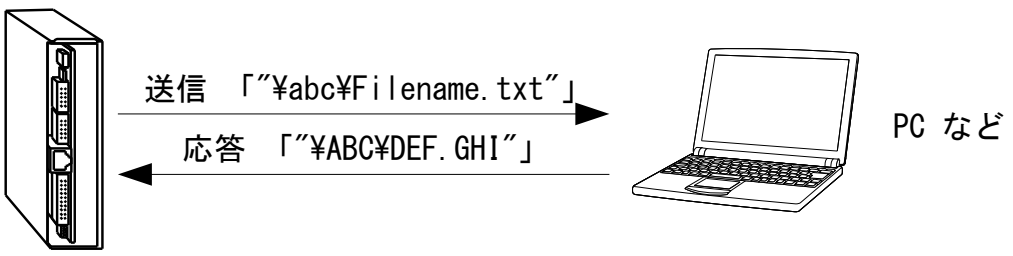
```
  SET_IP 192 168 0 20 255 255 255 0 192 168 0 248 /* 自己IP設定
  SET_IP UDP PACKET 61440                  /* UDP Port番号

  IP_CONV 192 168 0 211 OppdIp             /* 変数 OppdIp にOPPDのIP 192.168.0.211 を入れる
  IP_CONV 192 168 0 67 PcIp                /* 変数 PcIp にPCのIP 192.168.0.67 を入れる
  SET_DEST OppdIp UDP_XO$                  /* 宛先のIPと送信文字列変数 UDP_XO$ を紐付け。使用する文字列変数は SET_IP コマンドで確認
  PACKET UDP_XO$ &h4100 &h5701 &h0004 &h002A 1  /* UDP_XO$ に順々にバイナリWordデータとして格納→送信 (このコマンドはダイヤルロック)
  WAIT IPC(UDP_RO$)!=0                     /* 受信待ち

  C=0 :  D=0
  DO
*retry
    UDP_RO$=""                             /* 受信文字列変数クリア
  SET_DEST OppdIp UDP_XO$                  /* 宛先設定
  PACKET UDP_XO$ &h4100 &h5700 &h0008 &h000C C &h001F D  /* 輝度設定
  timer_=10
  WAIT IPC(UDP_RO$)!=0 OR timer_==0        /* 受信 または タイムアウト
  IF  timer_==0 THEN
    PRINT "No reply "
    END
  END_IF
  IP_CONV IPA(UDP_RO$)                     /* 送信元IP表示
  ptr_=UDP_RO$
  IF PTR(2,0)!=&h0041 THEN                 /* 応答確認
    TIME 100
    PRINT "retry"
    GOTO *retry
  END_IF
  C=C+70
  IF C>500 THEN
    C=0
  END_IF
  D=D+170
  IF D>500 THEN
    D=0
  END_IF

  DO
    UDP_RO$=""
    SET_DEST OppdIp UDP_XO$                /* 宛先設定
    PACKET UDP_XO$ &h4100 &h5200 &h0004 &h0039 &h0044 /* LAMP1 LAMP2 内部温度読み出し
    timer_=10
    WAIT (IPC(UDP_RO$)!=0)|(timer_==0)     /* 応答待ち
    IF timer_==0 THEN
      PR "TIME OUT OPPD"
    ELSE
      BREAK                                /* タイムアウトでなければ DO LOOP を抜ける
    END_IF
  LOOP
    ptr_=UDP_RO$
    PRINT "temp=" PTR(2,10) PTR(2,14)

    DO
      UDP_RO$=""
      SET_DEST PcIp UDP_XO$                /* 宛先設定(PC)
      PACKET UDP_XO$ &h0123 &h4567         /* 送信
      timer_=10
      WAIT (IPC(UDP_RO$)!=0)|(timer_==0)   /* 応答待ち
      IF timer_==0 THEN
        PR "TIME OUT PC"
      ELSE
        BREAK
      END_IF
    LOOP
    IP_CONV IPA(UDP_RO$)                   /* 送信元IP表示
    ptr_=UDP_RO$                           /* PC から [0]0x01 [1]0x23 [2]0x45 [3]0x67 と来る
    a1=PTR(2,0)                            /* [0]番目から Word 読み込み
    a2=PTR(1,2)                            /* [2]番目から Byte 読み込み
    a3=PTR(4,0)                            /* [0]番目から Long 読み込み
    PRINT "PC Data=" HEX$(a1) HEX$(a2) HEX$(a3)

  LOOP
```

■ 実行結果

```
RUN

#192 168 0 211            ← OPPD の IP Address
 temp= 0 0                ← 実験用 OPPD には照明が接続されていないので温度は0
192 168 0 67              ← PC の IP Address
 PC Data= 00000123 00000045 01234567 ← PC がループバックしたデータ
```

「¥」「″」の送信、受信(キャラクタ置換)

- MPCにとって ¥ や ″ は文字列を扱う際の特殊なキャラクタで、文字列の中でそれらを直接扱うことができません。
- ¥ を送信する場合は ¥¥ (または ¥x5c)、″ は ¥x22 とします。
- 受信文字列にそれらが含まれる場合は、他のキャラクタに置き換えます。
- 置き換えはパケット受信文字列 TCP_Rx$, UDP_Rx$ で可能です。



■ プログラム例

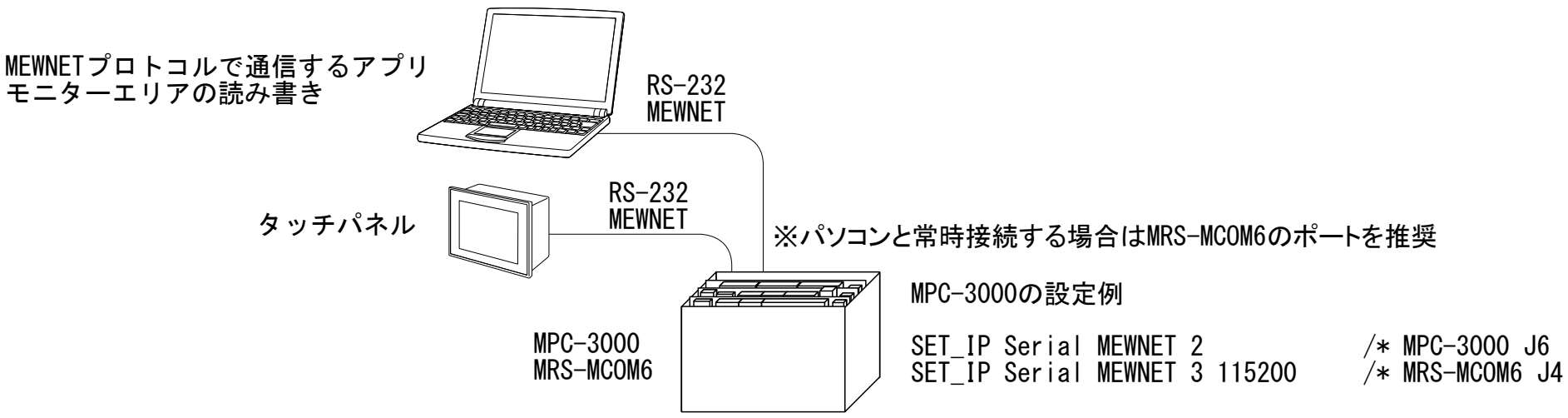
```
SET_IP 192 168 0 19 255 255 255 0 192 168 0 248
SET_IP TCP PACKET 9000 192 168 0 67        /* PCのIP

DO
  TCP_R0$=""                               /* 受信文字列クリア
  TCP_X0$="¥x22¥¥abc¥¥Filename.txt¥x22"    /* 「″¥abc¥Filename.txt″」と送信
  WAIT IPC(TCP_R0$)<>0                     /* 受信待ち
  SERCH TCP_R0$ &h22 &h27                  /* 「″¥ABC¥DEF.GHI″」と返ってくるので ″ を ' に置き換える
  SERCH TCP_R0$ &h5c &h20                  /* さらに ¥ をスペースに置き換える
  st$=TCP_R0$
  PR st$
LOOP
```

■ 実行結果

```
RUN

# ' ABC DEF.GHI'      ← st$
```

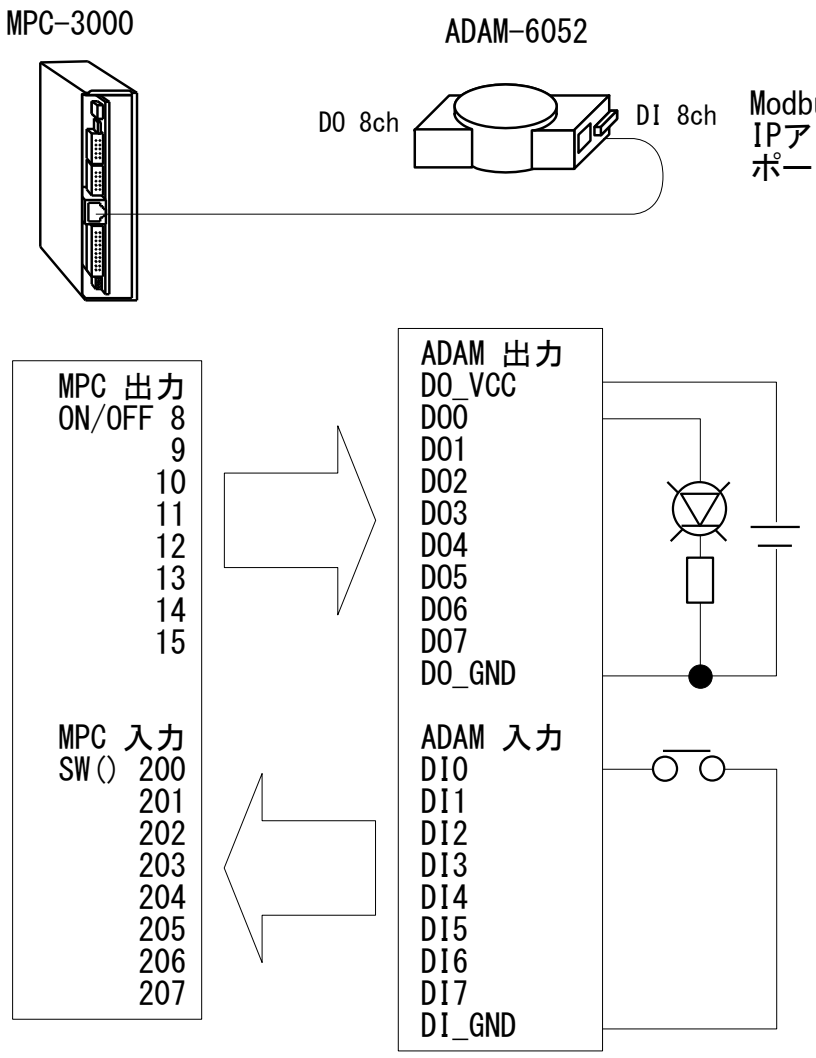
RS-232 MEWNET 接続

- MEWNETはEthernet接続ではありませんが、設定はSET_IPで行います。
- 設定を変更した場合は 「RUN IP_INIT」実行 → 電源再投入 です。
- サンプル作成:MPC-3000(SH2A) BL/I 2.01_23 2019/02/19
- MPC-3000/3200はRS-485,422非対応。(MRS-MCOM6でも非対応)

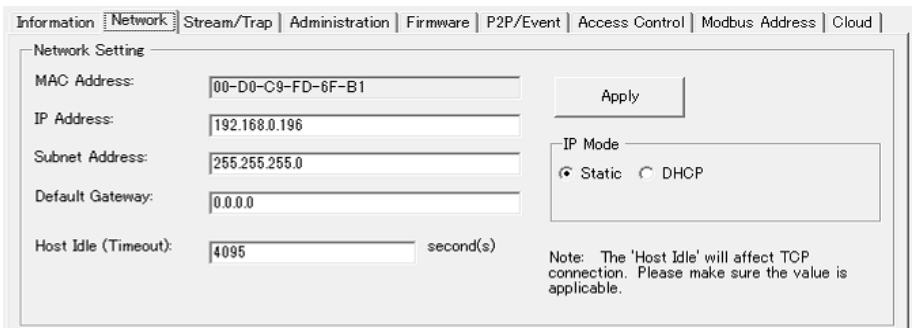


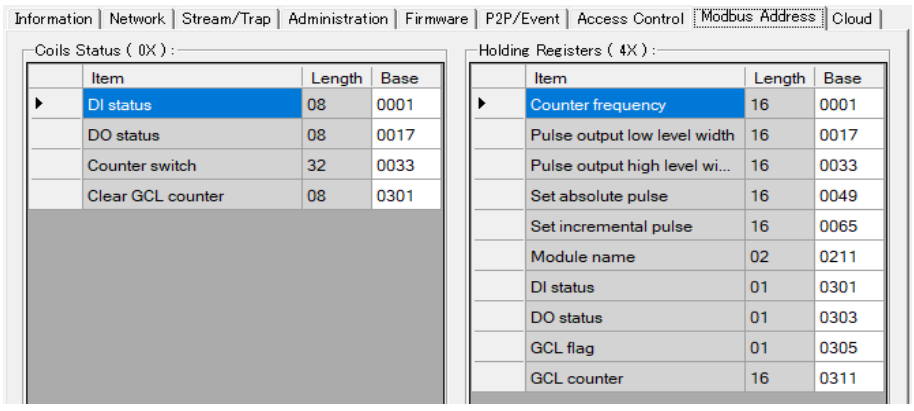
Advantech ADAM-6052 Modbus/TCP 接続

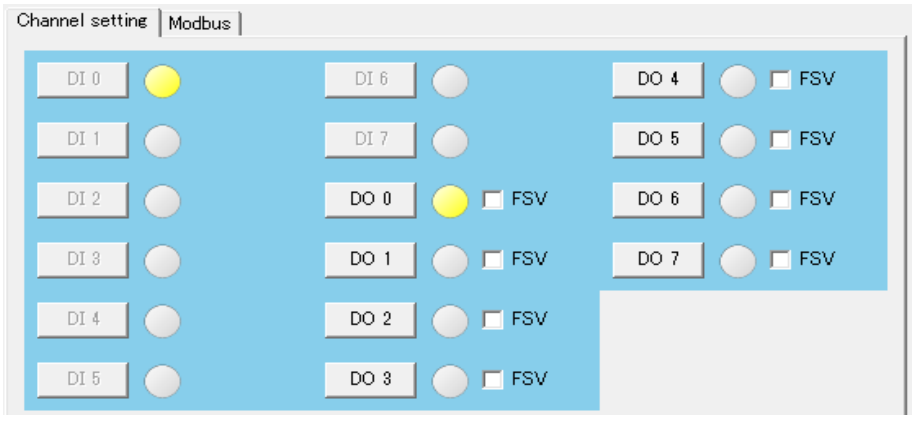
- ADAM-6052 は8入力/8出力のEthernet-based smart I/O です。
- MPC-3000はADAMとリンクするModbus/TCPプロトコルをサポートしています。
- バックグラウンドで常時実行され、MPCとADAMの入出力が共有されます。
- サンプル作成:MPC-3000(SH2A) BL/I 2.00_09 2018/11/08

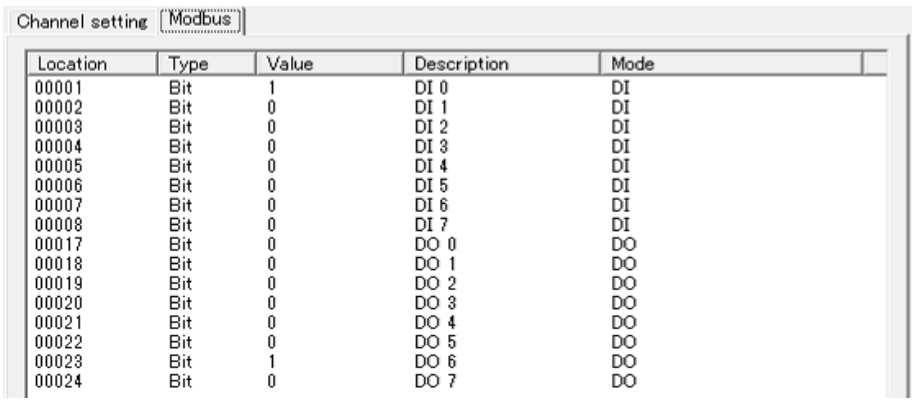


- Advantech Adam/Apax .NET Utility









■ プログラム例

```
SET_IP 192 168 0 20 255 255 255 0 192 168 0 248
SET_IP TCP Modbus ADAM 192 168 0 196

DO
  FOR i=8 TO 15
    ON i                      /* DO0~7 ON
    IF SW(200)==1 THEN        /* DI0から入力
      TIME 200
    ELSE
      TIME 500
    END_IF
    OFF i                     /* DO0~7 OFF
  NEXT
LOOP
```

■ SET_IP情報

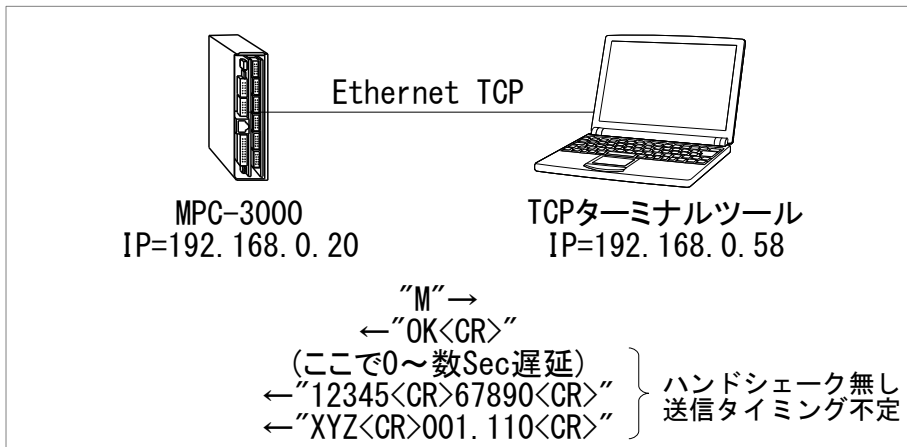
```
#SET_IP
IPadrs = 192.168.0.20
 MASK  = 255.255.255.0
Gateway= 192.168.0.248
OFF Delay ACK
MACadrs= 001EC08DAB6E

1 : =  TCP Modbus ADAM 502 Master 192.168.0.196 ref_time = 50
#
```

連続パケット受信と文字列分割例

- 複数パケットのデータを1文字列とし、INPUT_TCPの繰り返しによりターミネータで分割します。
- INPUT_TCPコマンドについては Webコマンドリファレンスを参照してください。
- TCP_XO$は受信バッファが空でなければ送信しません。必ず事前にクリアしてください。
- サンプル作成: MPC-3000(SH2A) BL/I 2.01_23 2019/02/28

■ MPCプログラム

```
SET_IP 192 168 0 20 255 255 255 0 192 168 0 248
SET_IP TCP PACKET 9000 PC 192 168 0 58
DIM a(10)

DO

  WHILE LEN(TCP_RO$)<>0      /* 受信バッファクリア
    INPUT_TCP 0 CLR_BUF
    TIME 50
  WEND

  TCP_XO$="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
    END_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
    END_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
  END_IF

  PR "---"
LOOP
```



このように、ハンドシェーク無しで複数の応答がある場合、パケットがくっついて1つの文字列になることがあり、TCP_RO$による受信では不都合。

このサンプルはPCから送信された文字列
data0<CR>data1<CR>data2<CR>...dataN<CR>
を<CR>区切りで読み込みます。

■ SET_IP情報

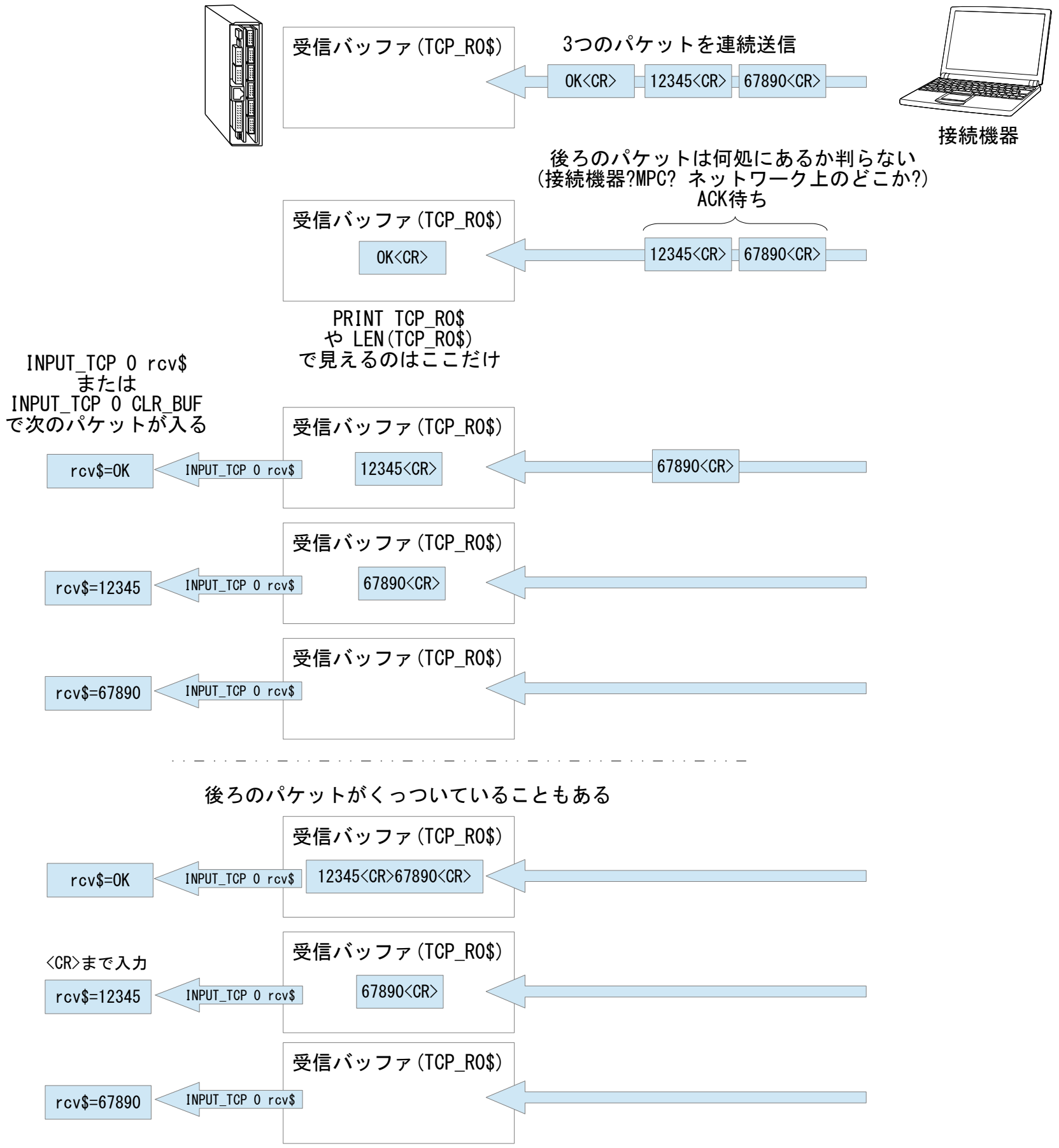
この数値がチャンネル番号

```
1 : =  TCP   PACKET   9000    <TCP_XO$><TCP_RO$> Master 192.168.0.58 ref_time = 50 PC
```

■ 実行結果

```
P$( 100 )= OK                <- OK 応答
P$( 101 )= 12345             <- 文字列
P$( 102 )= 67890
P$( 103 )= XYZ
P$( 104 )= 001.110
st$= OK 12345 67890 XYZ 001.110
a( 0 )= 12345                <- 数値 (GET_VAL は数値のみ配列変数に取り込む)
a( 1 )= 67890
a( 2 )= 1
a( 3 )= 110
a( 4 )= -2147483648
---
P$( 100 )= NG                 <- NG や ER 応答の場合
---
```

■ INPUT_TCP連続受信イメージ



■ MPCプログラム例

```
SET_IP 192 168 0 20 255 255 255 0 192 168 0 248
SET_IP TCP PACKET 9000 192 168 0 57
DO
  /* ネットワーク上に残っている全データをクリア
  WHILE LEN(TCP_RO$)<>0
    INPUT_TCP 0 CLR_BUF
    TIME 50              /* パケット受信に十分な時間
  WEND
  TCP_XO$="M"            /* コマンド送信
  WAIT TCP_RO$<>""       /* 受信待ち
  FOR i=0 TO 2           /* データ数:3個
    INPUT_TCP 0 EOL|13 rcv$
    PR rcv$
  NEXT
  TIME 50
LOOP
```

■ 実行結果

```
RUN

# OK
 12345
 67890
 OK
 12345
 67890
```

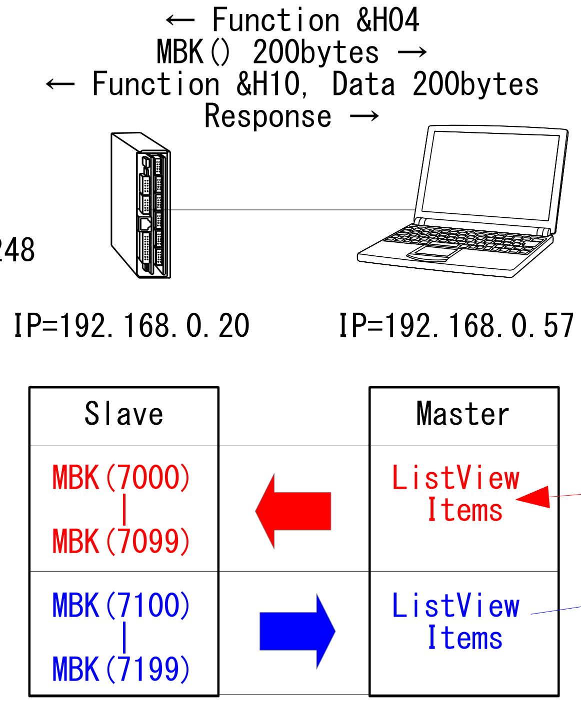
Modbus/TCPによるMPC～パソコン間メモリシェア

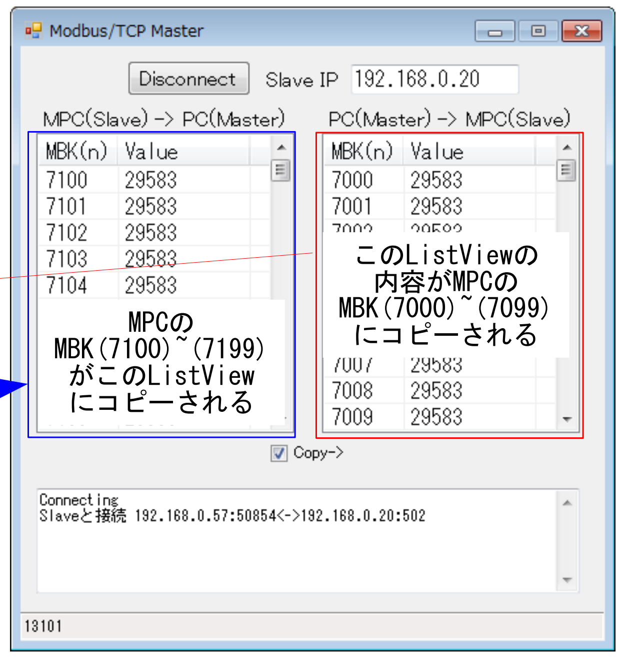
- MPC搭載のModbus/TCPを応用してパソコンとメモリシェアするVB.netアプリ例です。
- このアプリの対象はMBK(7000)～(7199)で固定です。
- サンプル作成:MPC-3000(SH2A) BL/I 2.01_23 2019/02/28
- Visual Studio 2013 VB.net

■ パソコンがMaster、MPCがSlave
パソコンの呼びかけに対してMPCが応答します。
通信のタイミングはパソコンに依存します。

・MPCプログラム例

```
SET_IP 192 168 0 20 255 255 255 0 192 168 0 248
SET_IP TCP Modbus

QUIT_FORK 1 *LED
j=0
DO
  FOR i=7100 TO 7199
    MBK(i)=j
  NEXT
  INC j
  TIME 100
LOOP

*LED
DO
  d=MBK(7000)
  OUT d 0
  WAIT d<>MBK(7000)
LOOP
```



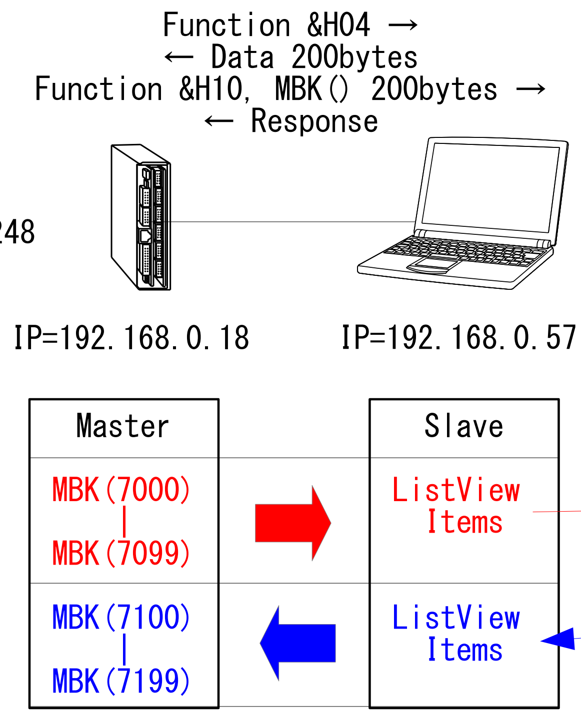


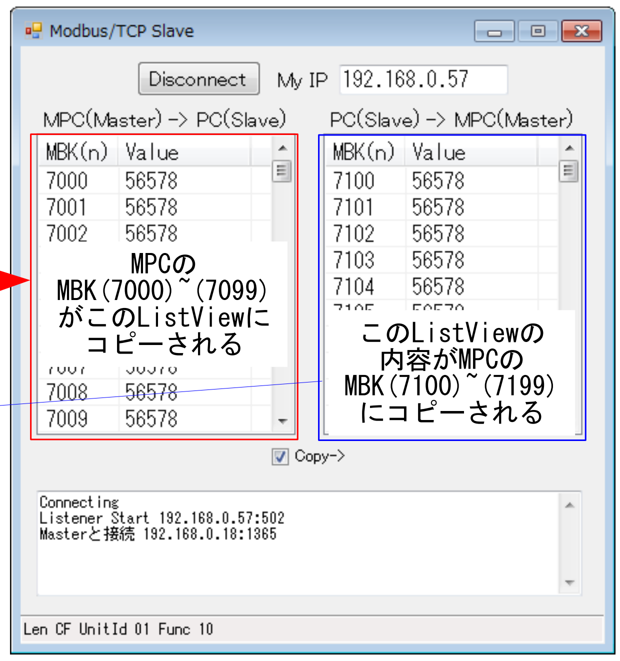
(Masterアプリ)

■ MPCがMaster、パソコンがSlave
MPCの呼びかけに対してパソコンが応答します。
MPCのバックグラウンドで常に実行されます。

・MPCプログラム例

```
SET_IP 192 168 0 18 255 255 255 0 192 168 0 248
SET_IP TCP Modbus SRC REF_T|20 192 168 0 57

QUIT_FORK 1 *LED
j=0
DO
  FOR i=7000 TO 7099
    MBK(i)=j
  NEXT
  INC j
  TIME 100
LOOP

*LED
DO
  d=MBK(7100)
  OUT d 0
  WAIT d<>MBK(7100)
LOOP
```





(Slaveアプリ)

■ MPC-3000 Modbus/TCP 通信例

・MasterからSlaveへ読み出し要求 (Function &H04)

```
00 00 00 00 00 06 01 04 00 1F 00 64
```

Slaveの応答

```
00 00 00 00 00 CB 01 04 C8 00 7B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00
```

・MasterからSlaveへ書き込み例 (Function &H10)

```
00 00 00 00 00 CF 01 10 00 1F 00 64 C8 00 00 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4
22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4
22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4
22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22 B4
22 B4 22 B4 22 B4 22 B4 22 B4 22 B4 22
```

Slaveの応答

```
00 00 00 00 00 06 01 10 1B 58 00 64
```

MPCにアクセスするModbus/TCPプロトコル例

- 任意のメモリエリアにアクセスします。
- サンプル作成:MPC-3000(SH2A) BL/I 2.01_24 2019/04/16

■ MPCのMBK(6000)～(6005) に "BADCFEHGT.TXT" を書き込む

【Query】

		Length		Func Code	Start Addr	Data Count	Byte	Data0	Data1	...						
00	00	00	00	00 13	01	10	17 70	00 06	0C	42 41	44 43	46 45	48	47	54	2E 54 58

【Response】

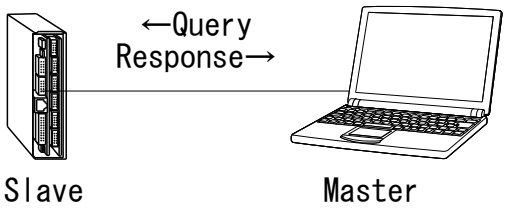
		Length		Func Code	Start Addr	Data Count
00	00	00	00	00 06	01	10	17 70	00 06

書き込み後、MPCで下記を実行してみます。

```
  FOR i=6000 TO 6005
    PRINT "MBK(" i ")=" HEX$(MBK(i))
  NEXT
  s$=MBK$(6000,12)
  PRINT "s$=" s$
```

実行結果

```
 MBK( 6000 )= 00004241
 MBK( 6001 )= 00004443
 MBK( 6002 )= 00004645
 MBK( 6003 )= 00004847
 MBK( 6004 )= 0000542E
 MBK( 6005 )= 00005458
 s$= ABCDEFGH.TXT
```

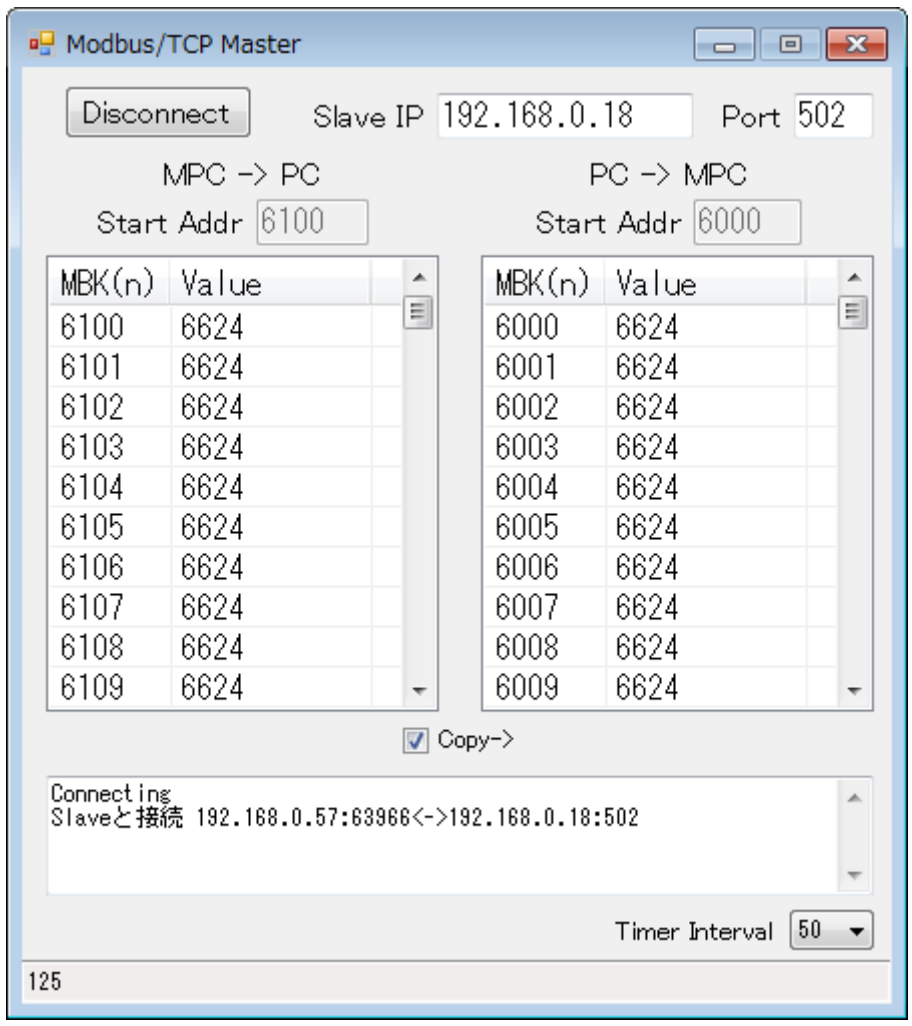


・MPCの宣言例

```
SET_IP TCP Modbus     /* Port=502
SET_IP TCP Modbus 501 /* Port=501
```

・Master側 Function Code

MBK書き込み &H10
MBK読み込み &H03
I/Oオン/オフ &H05
I/O読み込み &H01



任意のMBKエリアを100wordずつ読み書きするサンプル。
Port番号を変えれば複数のPCからアクセス可。

■ MPCのMBK(6000)～(6005) を読み出す

【Query】

		Length		Func Code	Start Addr	Data Count
00	00	00	00	00 06	01	03	17 70	00 06

【Response】

		Length		Func Code	Byte	Data0	Data1	...						
00	00	00	00	00 0F	01	03	0C	42 41	44 43	46 45	48	47	54	2E 54 58

■ MPCのI/O 70000 をビットオン

【Query】

		Length		Func Code	Bit No	Data
00	01	00	00	00 06	01	05	58 00	FF 00

■ MPCのI/O 70000 をビットオフ

【Query】

		Length		Func Code	Bit No	Data
00	01	00	00	00 06	01	05	58 00	00 00

■ MPC I/O 70000～16ビット読み出し

【Query】

		Length		Func Code	Start Addr	Data Count
00	01	00	00	00 06	01	01	58 00	00 10

【Response】

		Length		Func Code	Byte	Data		
00	01	00	00	00 05	01	01	02	34 12

← 事前にMPC側は OUT &H1234 70000~Wrd とした。

■ MBK(5000)～MBK(5099) を読み出す

事前にMPCでプログラムを実行してデータ作成

```
FOR i=5000 TO 5099
  MBK(i)=i
NEXT
```

【Query】

開始アドレス x1388=5000　読み出し数 x0064=100word

00 00 00 00 00 06 01 03 13 88 00 64

【Response】

MBK(5000)=5000

00 00 00 00 00 CB 01 03 C8 13 88 13 89 13 8A 13 8B 13 8C 13 8D 13 8E 13 8F 13 90 13 91 13 92 13 93 13 94 13 95 13 96 13 97 13 98 13 99 13 9A 13 9B 13
9C 13 9D 13 9E 13 9F 13 A0 13 A1 13 A2 13 A3 13 A4 13 A5 13 A6 13 A7 13 A8 13 A9 13 AA 13 AB 13 AC 13 AD 13 AE 13 AF 13 B0 13 B1 13 B2 13 B3 13 B4 13
B5 13 B6 13 B7 13 B8 13 B9 13 BA 13 BB 13 BC 13 BD 13 BE 13 BF 13 C0 13 C1 13 C2 13 C3 13 C4 13 C5 13 C6 13 C7 13 C8 13 C9 13 CA 13 CB 13 CC 13 CD 13
CE 13 CF 13 D0 13 D1 13 D2 13 D3 13 D4 13 D5 13 D6 13 D7 13 D8 13 D9 13 DA 13 DB 13 DC 13 DD 13 DE 13 DF 13 E0 13 E1 13 E2 13 E3 13 E4 13 E5 13 E6 13
E7 13 E8 13 E9 13 EA 13 EB ←MBK(5099)=5099

■ MBK(6000)～MBK(6099) に 6000～6099を書き込み

【Query】

開始アドレス x1770=6000　書き込み数 x0064=100word　6000

00 00 00 00 00 CF 01 10 17 70 00 64 C8 17 70 17 71 17 72 17 73 17 74 17 75 17 76 17 77 17 78 17 79 17 7A 17 7B 17 7C 17 7D 17 7E 17 7F 17 80 17 81 17
82 17 83 17 84 17 85 17 86 17 87 17 88 17 89 17 8A 17 8B 17 8C 17 8D 17 8E 17 8F 17 90 17 91 17 92 17 93 17 94 17 95 17 96 17 97 17 98 17 99 17 9A 17
9B 17 9C 17 9D 17 9E 17 9F 17 A0 17 A1 17 A2 17 A3 17 A4 17 A5 17 A6 17 A7 17 A8 17 A9 17 AA 17 AB 17 AC 17 AD 17 AE 17 AF 17 B0 17 B1 17 B2 17 B3 17
B4 17 B5 17 B6 17 B7 17 B8 17 B9 17 BA 17 BB 17 BC 17 BD 17 BE 17 BF 17 C0 17 C1 17 C2 17 C3 17 C4 17 C5 17 C6 17 C7 17 C8 17 C9 17 CA 17 CB 17 CC 17
CD 17 CE 17 CF 17 D0 17 D1 17 D2 17 D3 ←6099

【Response】

00 00 00 00 00 06 01 10 17 70 00 64

MMC(Multi Media Card)アクセス

- Ethernet機能ではありませんが、MMCの読み書き例です。
- USBメモリはシークタイムが遅く、点データ、MBKデータの一括保存/読み出しには不向きです。
- サンプル作成:MPC-3000(SH2A) BL/I 2.01_43 2020/08/25

■ 点データ、MBKデータの保存・読み出しプログラム例

```
ON_ERROR  *sys_err

FILL MBK(100) 7000 1 1
FILL P(100) 7000 1 1

IF MMC(0)<>1 THEN                    /* 書き込む前は必ずカード有無確認
  PR "MMC None"
  END
END_IF
SYSCLK=0
MMC_PSAVE MBK(100) 7000  "Mbk.MBK"  /* 既存ファイルには上書き
PR "MBK SAVE" SYSCLK "msec"
SYSCLK=0
MMC_PSAVE P(100) 7000  "Point.P2K"
PR "POINT SAVE" SYSCLK "msec"
TIME 1000
SYSCLK=0
MMC_PLOAD "Mbk.MBK"                  /* ※ ファイルが無いとき、カードが無いときはエラー
PR "MBK LOAD" SYSCLK "msec"
SYSCLK=0
MMC_PLOAD "Point.P2K"
PR "POINT LOAD" SYSCLK "msec"
END

*sys_err
  PRINT   "ERROR Line:" err_&&H00FFFFFF "Code:" err_>>24 "Message:" ERR$(err_)
  END
```

■ 実行結果

```
RUN

# MBK SAVE 1728 msec
 POINT SAVE 3012 msec
MMC_Card inserted Mounted
 MBK LOAD 602 msec
MMC_Card inserted Mounted
 POINT LOAD 1048 msec
FILES
MMC_Card inserted Mounted

     MBK.MBK      2020/ 9/ 1 14:30 123993
   POINT.P2K      2020/ 9/ 1 14:30 218672
#
```

※で存在しないファイルを指定した場合

```
RUN

# MBK SAVE 1756 msec
 POINT SAVE 2992 msec
MMC_Card inserted Mounted
 ERROR Line: 160 Code: 87 Message: MMCカードにファイルがありません
```

■ 1行書き込み・読み出しプログラム例

```
IF MMC(0)<>1 THEN              /* MMCチェック
  PR "MMC Error"
  END
END_IF
FILE$="LOG.TXT"                /* ファイル名 FILE$は予約変数
REMOVE FILE$                   /* 既存ファイル消去
FOR c=0 TO 19
  tm$=TIME$(1)+" "+STR$(c)
  MMC_WRITE tm$                /* 追記
  TIME 500
NEXT
TIME 1000
DO
  MMC_READ a$                  /* 1行読み出し
  PR a$
  IF EOF(1)==0 THEN
    BREAK
  END_IF
LOOP
PR "Complete"
```

■ 実行結果

```
RUN

# 14:31:46 0      ← PR a$ の表示
 14:31:46 1
 14:31:47 2
(中略)
 14:31:56 18
 14:31:56 19

 Complete
#FILES           ←ファイル一覧表示
MMC_Card inserted Mounted

    LOG.TXT 2019/ 3/28 14:31 250
#CAT FILE$       ←ファイル内容表示
14:31:46 0
14:31:46 1
14:31:47 2
(中略)
14:31:56 18
14:31:56 19
#MMC_RENAME "LOG.TXT" "OLD_LOG.TXT" ←ファイル名変更
#FILES
MMC_Card inserted Mounted

OLD_LOG.TXT 2019/ 3/28 14:31 250
#
```

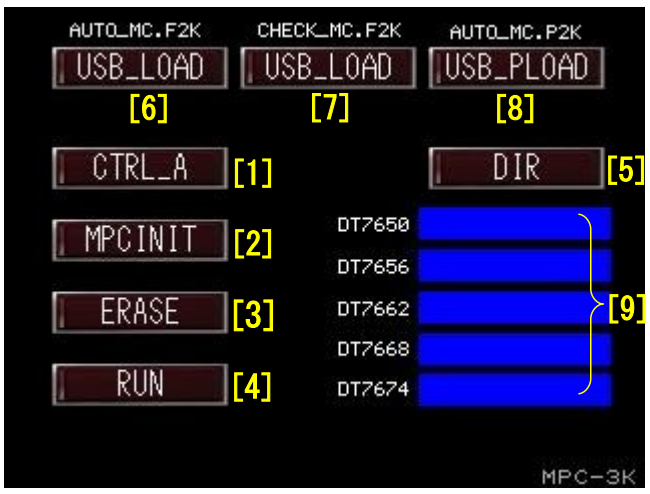
タッチパネルコマンド

- タッチパネルに配置したボタンでコマンドを実行します。USBからのプログラム更新・保守に利用できます。
- ワードアドレス DT7802 (32bit) に定数を書き込むとコマンドを実行します。コマンド実行中は DT7803,DT7802 が &Hnnnn80xx となります。USB_LOAD、USB_PLOAD実行中はnnnnが1行ずつインクリメントされ終了すると&H00000000になります。USB_LOAD,USB_PLOAD以外は直ちに&H00000000となります。実行中の途中停止はできません。
- ファイル名は固定です。データはMPCから保存したものを用いてください。
- 読込は時間がかかります。300Kbyte、11000行のプログラムで約10分(使用メモリにより異なります)
- サンプル作成: MPC-3000(SH2A) BL/I 2.01_24 2019/04/10

	コマンド	機能		DT7802定数	bit
1	CTRL+A	停止	USB_LOAD、USB_PLOAD中無効	&HFFFFFF01 (-255)	0
2	MPCINIT	SRAM初期化	プログラム、点データ、MBKデータクリア	&HFFFFFF02 (-254)	1
3	ERASE	FROM初期化	FROMのプログラムクリア	&HFFFFFF04 (-252)	2
4	RUN	プログラム実行	LOAD後はFROM書き込み後実行	&HFFFFFF08 (-248)	3
5	DIR	USBファイル一覧作成	DT7650～。12byte/1file。20fileまで	&HFFFFFF10 (-240)	4
6	USB_LOAD	USBメモリからプログラム読込	ファイル名 "AUTO_MC.F2K"	&HFFFFFF20 (-224)	5
7	USB_LOAD	USBメモリからプログラム読込	ファイル名 "CHECK_MC.F2K"	&HFFFFFF40 (-192)	6
8	USB_PLOAD	USBメモリから点データ読込	ファイル名 "AUTO_MC.P2K"	&HFFFFFF80 (-128)	7

■ デザイン例 (Pro-face GP4301)
ボタン([5]以外)押下で読込中や完了を表す別の画面に切り替わります。
[5]を押すと下にファイル名を表示します(8.3文字)。

メインの画面



[1]～[8] スイッチ/ランプ

スイッチ機能
(ﾜｰﾄﾞｽｲｯﾁと画面切替ｽｲｯﾁのマルチファンクション)
ワードスイッチ
ワードアドレス: DT7802
ワード動作: データ書き込み
データ形式: Hex
ビット長: 32ビット
定数: (上記表)
画面切替スイッチ
画面切替動作: 画面切替
画面番号: [1]～[4] 2004、[6]～[8] 2005

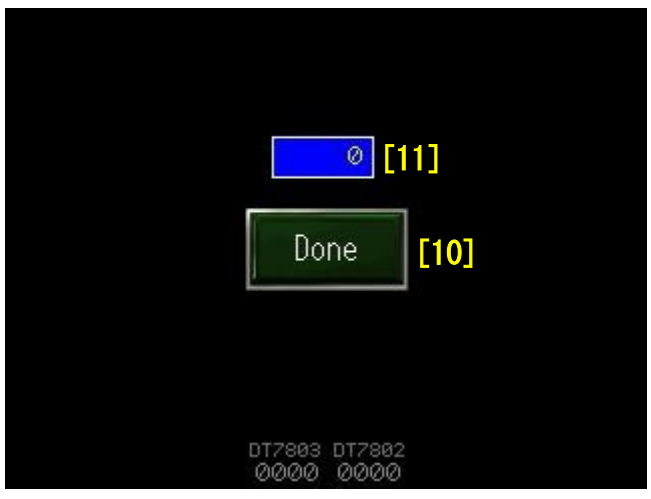
ランプ機能
ビットアドレス: DT007802.0~7

[9] データ表示器

表示データ: 文字列表示
モニタワードアドレス: DT7650 から 6ワード間隔
表示文字列数: 12

[1]～[4]実行中・完了時の画面
(画面番号2004)

[6]～[8]ファイル読込中・完了時の画面
(画面番号2005)



[10] スイッチ/ランプ

スイッチ機能
画面切替スイッチ
画面切替動作: 前画面に戻る

スイッチ共通
インターロック機能
アドレスを使用する
インターロックアドレス: DT007802.F
タッチ有効条件: ビットOFF時有効

ランプ機能
ビットアドレス: DT007802.F
銘板: OFF時 Done、ON時 Wait..

[11] データ表示器

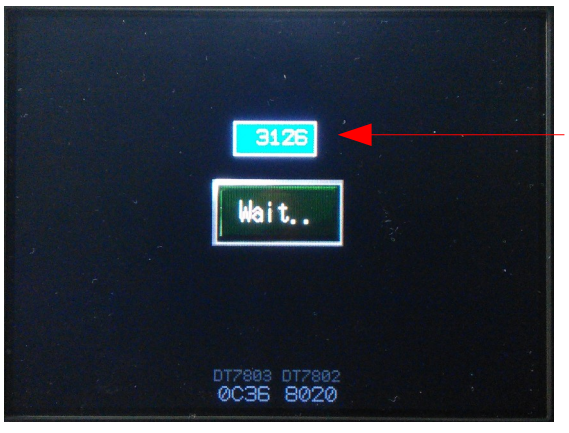
表示データ: 数値表示
モニタワードアドレス: DT007803
データ形式: 16ビット Dec

実機動作画面

DIR押下で表示

[1]～[4]完了時

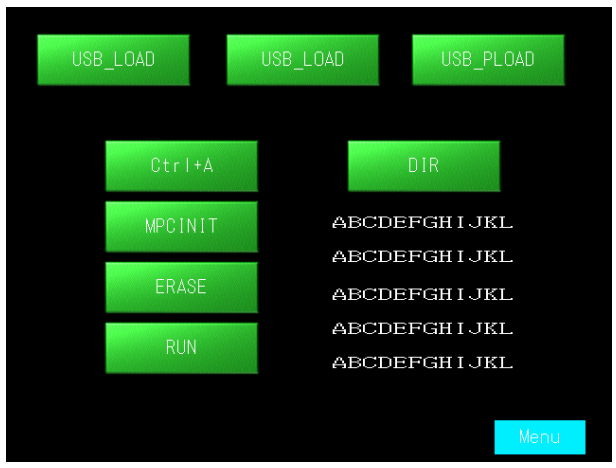
[6]～[8]ファイル読込中
終了すると Done になります



読込中1++
終わると0

■ デザイン例 (KEYENCE VT3-V10)
コマンドボタンを押すと、実行中・完了画面に替わります(DIR除く)。実行中はボタンは非表示となります。

メインの画面



実行中・完了の画面

(LOAD・DIR以外)

(LOAD)

・コマンドのボタンは
スイッチ
　ワードセット
　　スイッチモード: データセット
　　対象ワードデバイス: PLC DT007802 2ワード
　　セットデータ: (前記表)
　ページ切替(DIR除く): ページNo指定

・DIRのファイル名表示は
文字表示
　対象ワードデバイス: PLC DT007650～ 6ワードずつ
　　上位/下位バイト入替
　表示桁数: 12

・Doneボタンは
Nステート部品
　設定
　　状態数: 1
　　指定方法: ワードデバイス
　　参照デバイス: PLC DT007802
　　範囲外指定時: 非表示
　状態0
　　部品の種類: スイッチ・ランプ
　　設定
　　ランプスイッチ
　　スイッチビットデバイス: 無効
　　ランプビットデバイス: 演算 DT007802='0'
　　ページ切替: ページNo指定

・LOAD中のステップ表示は
数値表示
　参照デバイス: PLC DT007803 1ワード 符号無しバイナリ

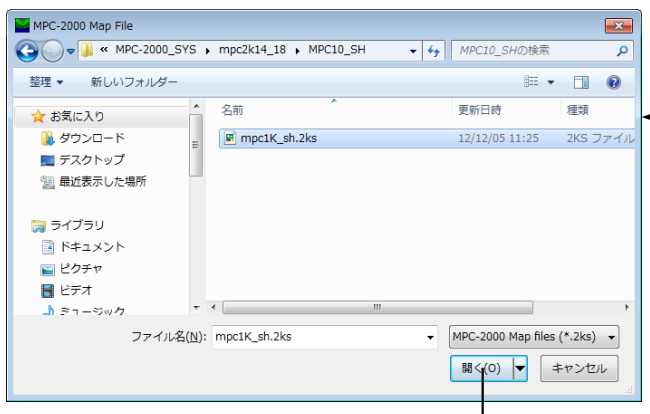
ファームウェアのUpdate手順

- ファームウェアのUpdateはシステムローダ sysld2k.exeで行います。
- sysld2k.exe は Accel_Setup_jpn.msi で C:¥Program Files¥Accel または C:¥Program Files (x86)¥Accel にインストールされます。
- FTMW2K の [System Loader] ボタン押下、または sysld2k.exe を直接起動してください。

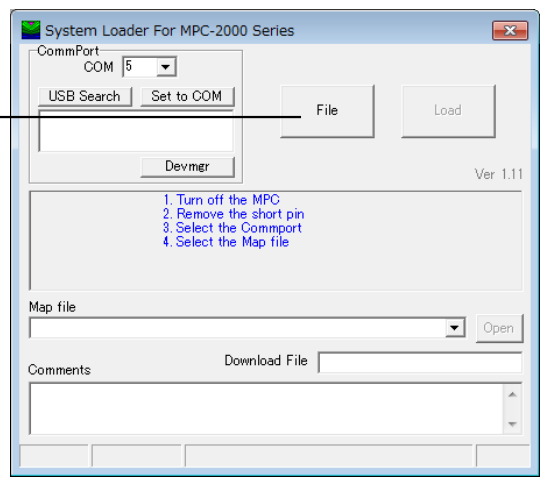
■ 最新ファームウェアはホームページからダウンロードできます。
https://accelmpc.co.jp/downloads/firmeware.html

該当機種のZIPをダウンロードして展開して下さい。
mpc3000.2ks
MPC3000.S

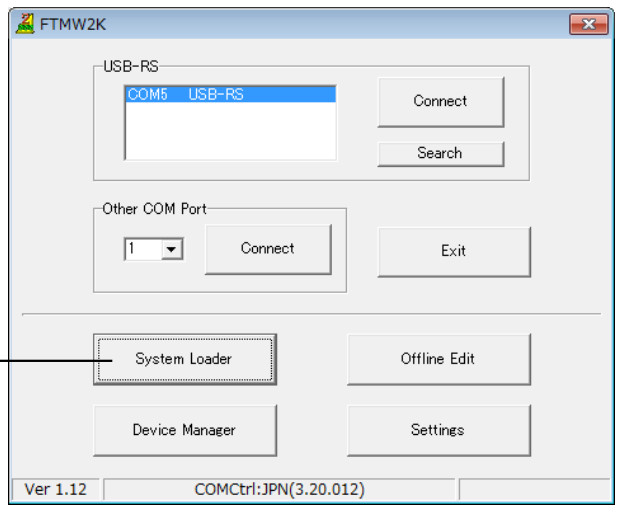
■ アップデート手順
MPCの電源を切りショートピンSP1を抜きます。
パソコンとMPCをプログラミングケーブルで接続します。



展開したフォルダ内の
mpc3000.2ksを選択します。

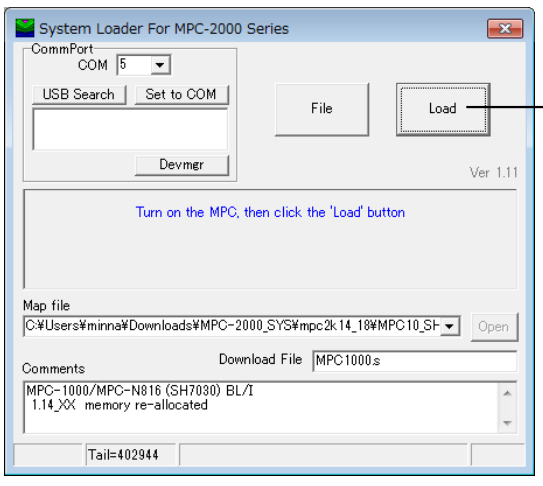


COMポートを確認して下さい。
File ボタンを押します。

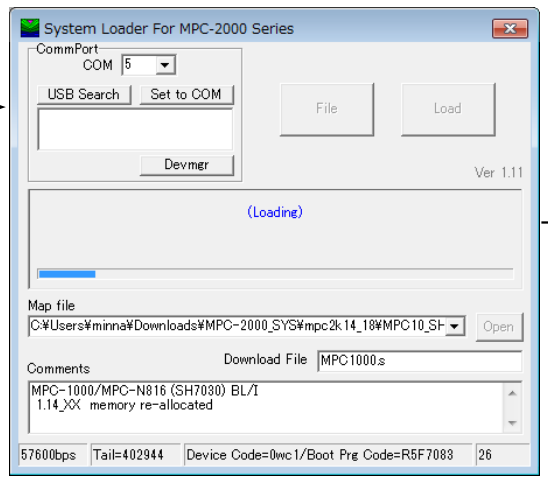


FTMW2Kの System Loader ボタンで
sysld2k.exeが起動します。

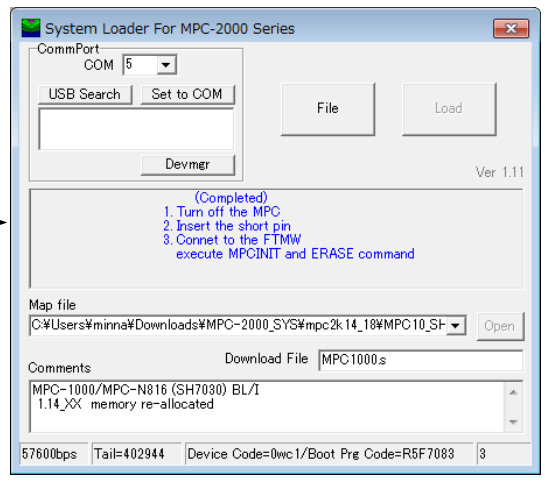
直接起動してもOkです。



MPCの電源を入れて
Loadボタンを押します。



MPCへ転送中。

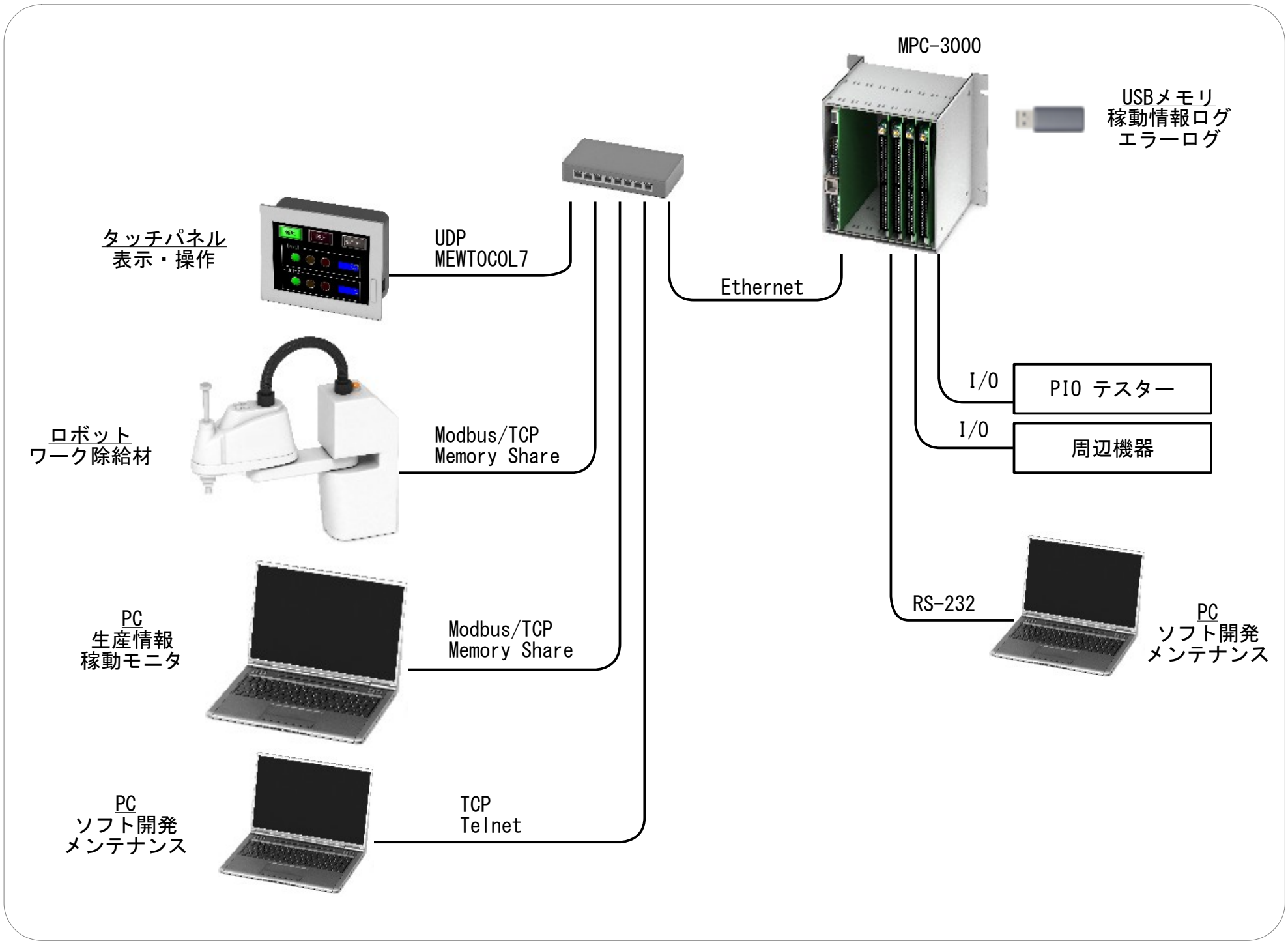


転送が完了したら
電源を切りショートピンを戻します。

アプリケーション実例

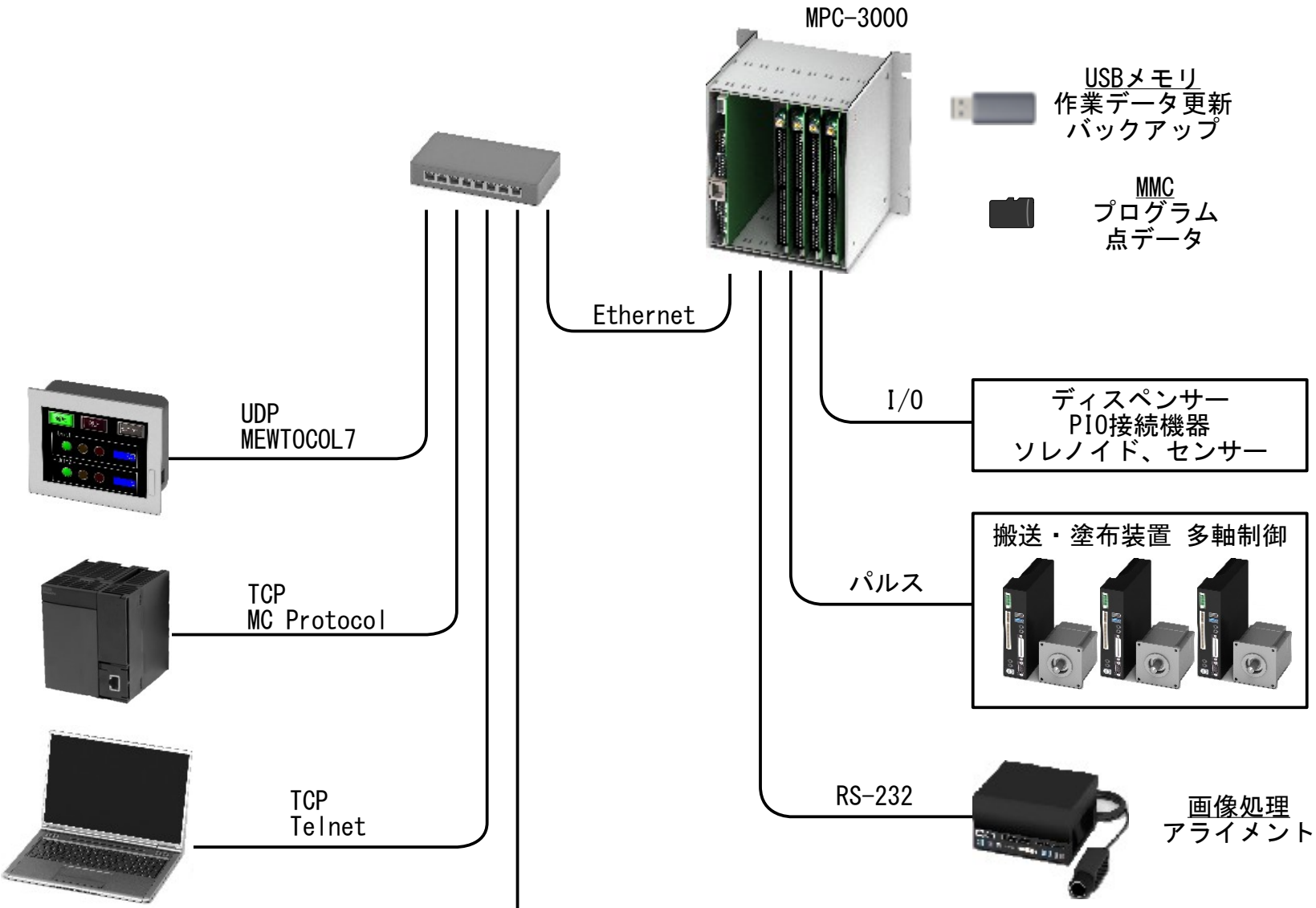
（装置メーカー様のご協力・承諾を得て主要機器の構成を掲載しています）

■ 電子部品テスターハンドラー



■ 大型塗布ライン

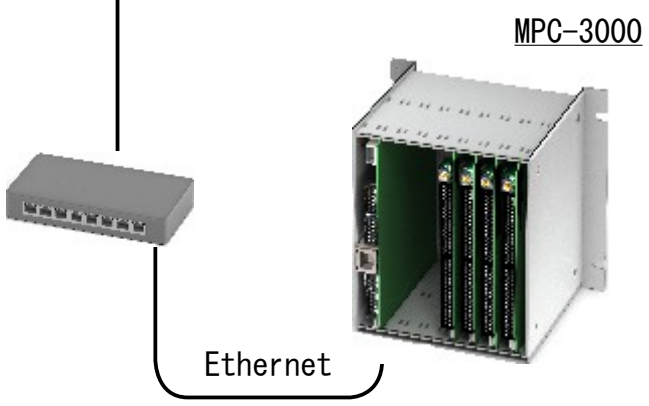
上流側装置



MPC間はModbus/TCP Memory Share

下流側装置

上流側装置とほぼ同仕様



■ 微細部品組立加エライン

上流側装置

MPC-3000

タッチパネル
表示・操作

UDP
MEWTOCOL7

Ethernet

スカラーRB
搭載・供給

Modbus/TCP
Memory Share

パルス

サーボモータ
ターンテーブル

I/O

パーツフィーダー
供給

CUnet
MPC間の
Memory Share

I/O

ソレノイド
センサー
PIO接続機器

下流側装置

MPC-3000

タッチパネル
表示・操作

UDP
MEWTOCOL7

Ethernet

スカラーRB
搭載・供給

Modbus/TCP
Memory Share

I/O

PIO 単軸ユニット
搬送

スカラーRB
搭載・搬送

Modbus/TCP
Memory Share

I/O

PIO サーボプレス
加工

I/O

ソレノイド
センサー
PIO接続機器