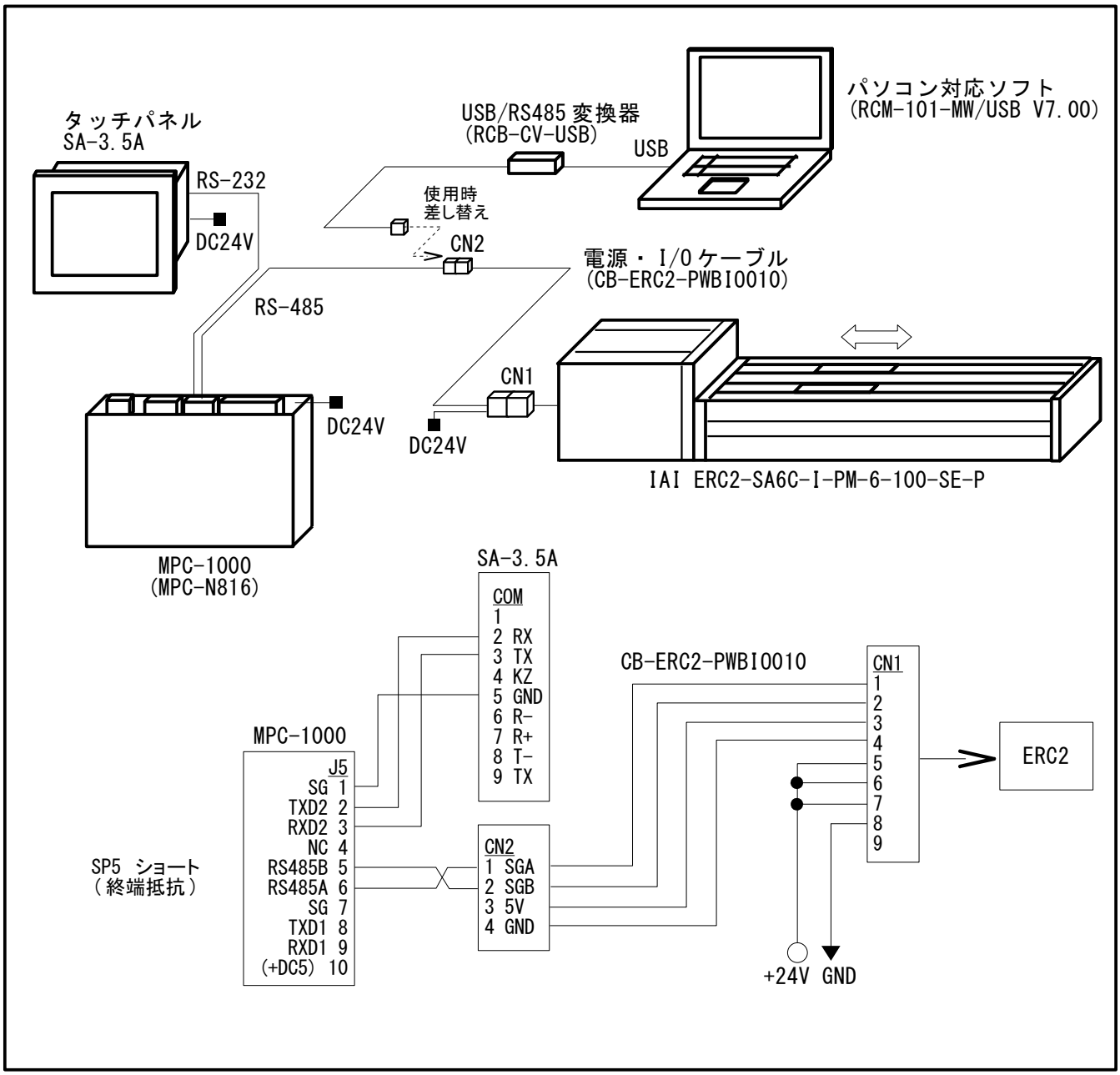
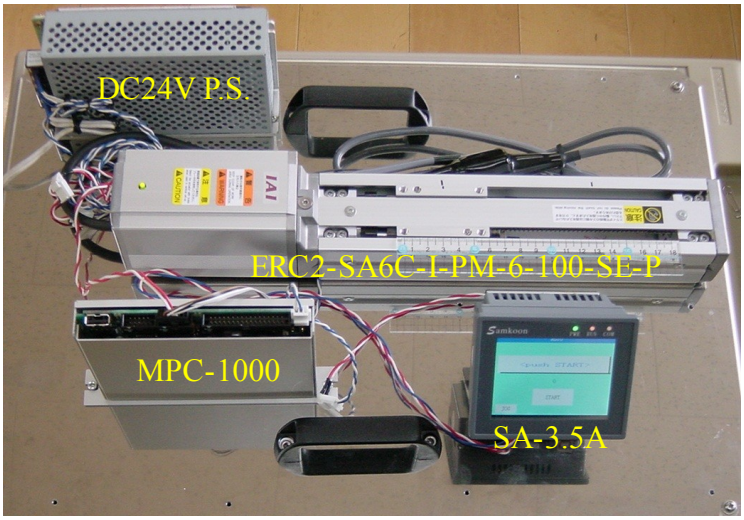
Application Note	資料作成　110405	資料番号 an2k-029
テーマ	㈱IAI 製 ロボシリンダ制御	
使用機器	MPC-1000/MPC-N816、IAI ERC2-SA6C-I-PM-6-100-SE-P、接続ケーブル 他	

機器構成.....1
概要.....3
ロボシリンダ ERC2-SA6C について.....3
参考資料.....3
その他.....3
タッチパネル画面と動作.....4
MPC サンプルプログラムで使用しているクエリ.....6
MPC サンプルプログラム サブルーチンの説明【 Modbus ASCII 】.....6
MPC サンプルプログラム【 Modbus ASCII 】.....7
MPC サンプルプログラム サブルーチンの説明 【 Modbus RTU 】.....15
MPC サンプルプログラム【 Modbus RTU 】.....15

機器構成

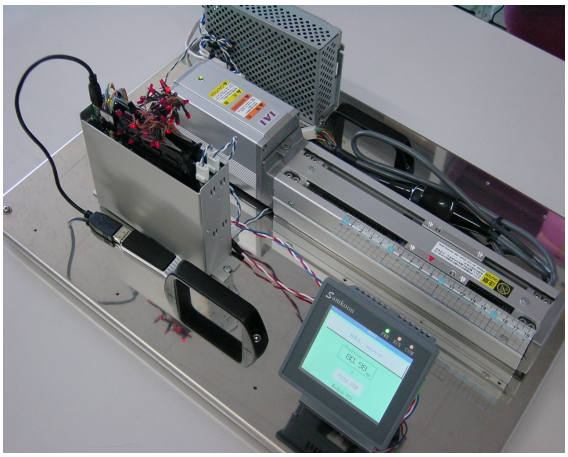


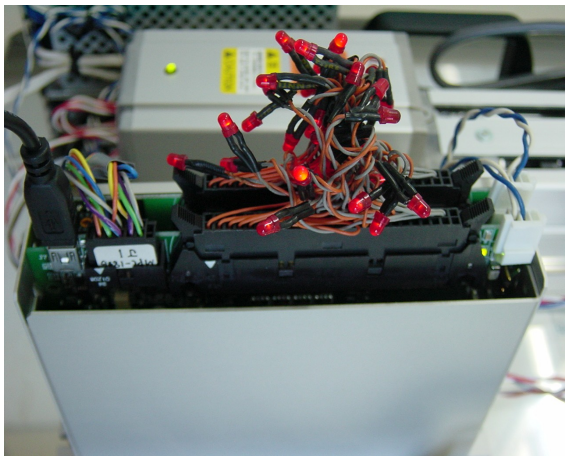
- MPC-1000 で制御



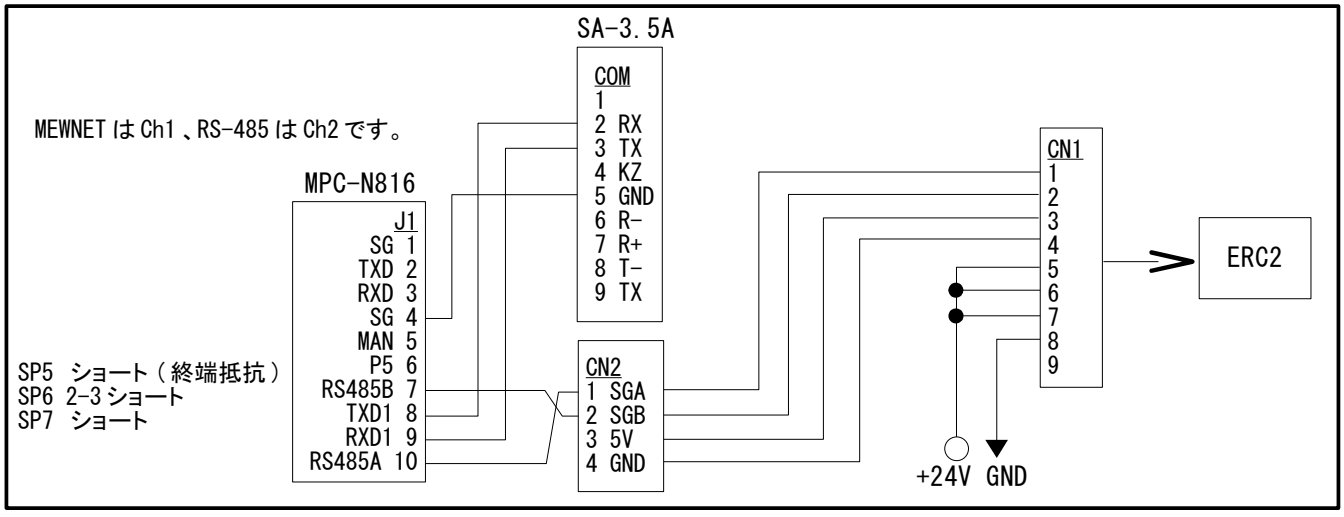
- MPC-N816 で制御

MPC-N816 と MIO-N816 の I/O コネクタは MPC-816 シリーズとコンパチ、プログラム言語は MPC-2000 シリーズコンパチです。





MPC-N816 と MIO-N816
(写真はチェック用の LED 付きコネクタを装着しています)



- メイン CPU が MPC-2000、MPC-2100L の場合は MRS-MCOM で RS-485 通信を行います。

概要

- ロボシリンダ(以下 ERC2)を RS-485 で制御します。
- 原点復帰、ERC2 内部のポジション(点)を使った移動、絶対座標移動、相対座標移動、現在位置読み取りなどを行います。
- タッチパネルで ERC2 内部ポジションのティーチング、パラメータ(位置、最高速度、加速度、減速度)の変更が行えます。

ロボシリンダ ERC2-SA6C について

- コントローラ一体、スライダタイプ、本体幅 58mm、パルスモータ(サーボ)
- このサンプルに使用したのは 100mm ストローク、最高速度 300mm/s、最大 16 軸接続、ポジション 64 点
- 通信方式には Modbus ASCII モードと Modbus RTU モードがあり ERC2 が自動判別します。
  前者は ASCII コード文字列で送受信、エラーチェックは LRC 方式。後者はバイナリコードで送受信、エラーチェックは CRC16 方式です。

  [送信文字列例] HOME クエリ “0105040BFF00 + チェックコード” の場合
  ASCII では 3A 30 31 30 35 30 34 30 42 46 46 30 30 45 43 0D 0A (先頭に：、末尾に CRLF を付加)
  RTU では 01 05 04 0B FF 00 FC C8
  という文字列になります。

参考資料

ERC2-SIO(MJ0159-5A).pdf ERC2 コントローラ(SIO 専用)一体型アクチュエータ 取扱説明書第 5 版
SERIAL-COMMUNICATION_MODBUS(MJ0162-3B).pdf ｼﾘｱﾙ通信【Modbus 版】取扱説明書 第 3 版
その他 カタログ等

その他

- ERC2 動作確認と通信レートの設定
  最初の動作確認と MPC との通信レート設定は「RC 用パソコン対応ソフト」で行いました。
  パラメータ
  SIO 通信速度「bps」：38400
  その他：初期値のまま
  軸 No は初期値=0 です。この場合スレーブアドレスは 1 です。

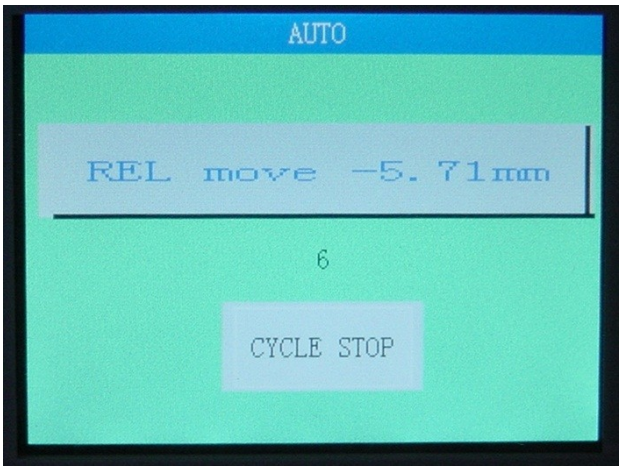
- ロボシリンダ関係購入品
  本体： ERC2-SA6C-I-PM-6-100-SE-P
  コントローラ： RCM-101-USB (パソコンソフト CD-ROM、取り説 CD-ROM、RCB-CV-USB(USB/RS422 変換器)、ケーブル、登録カード、他 のセット)
  電源・I/O ケーブル：CB-ERC2-PWBIO010 (ケーブル長 100mm)

- 中継コネクタについて
  図中の中継コネクタ CN2 は購入時は日圧のコネクタですが、PC と MPC を差し替えて使用するため AMP のコネクタに付け替えました。

- タッチパネルについて
  このサンプルでは中国製を用いていますが、デジタル、パナソニック、三菱、キーエンスでも同様の操作画面を作成でます。

- デバッグ用通信モニタ
  空き RS-232 ポートを用いて通信内容をモニタします。サンプルプログラムではポート番号を変数 MONI_PORT に設定します。タッチパネルを使用するときは MONI_PORT を 0 にします。モニタ方法の詳細はアプリケーションノート an2k-025.pdf をご参照ください。

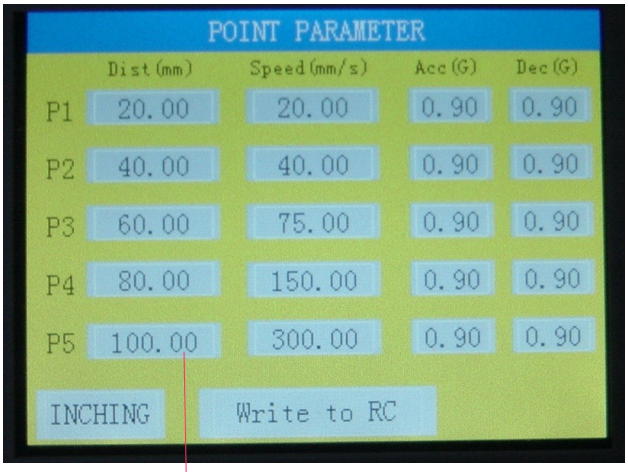
タッチパネル画面と動作



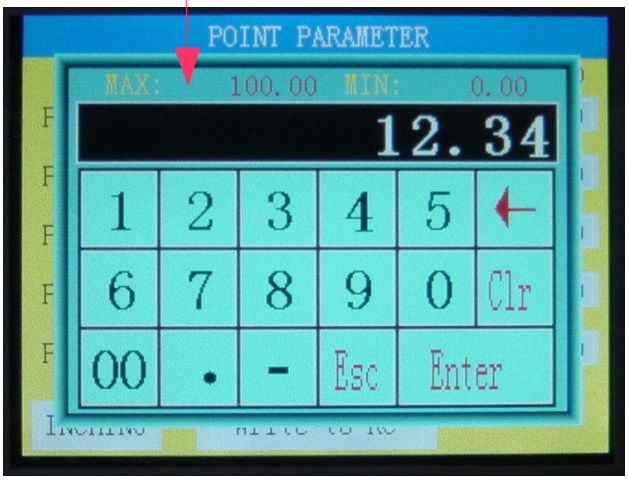
・AUTO
原点復帰
ERC2 内部ポジション移動 P1→P2→P3→P4→P5
絶対座標移動 0.00mm→100.00mm
相対座標移動 -10.00mm 3 回×2 回、-5.71mm7 回
を繰り返します。

ERC2 内部ポジション移動のスピード、加速度等は各ポジションのパラメータに依存します。
絶対座標移動、相対座標移動のスピード、加減速はプログラムで直接指定しています。

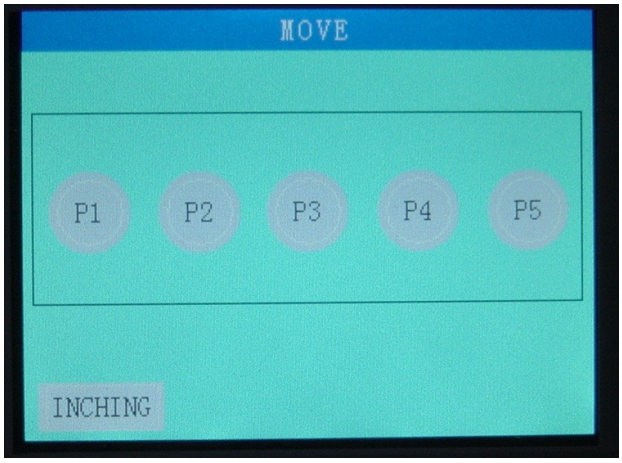
・INCHING
＋,－ボタンで Dist で選択した量だけ寸動します。
Position は現在位置です。
PRAM ボタンで内部ポジションパラメータ表示画面、
MOVE ボタンで内部ポジション移動画面、
TEACH ボタンで内部ポジション教示画面へ遷移します。
AUTO ボタンで AUTO 画面に戻ります。



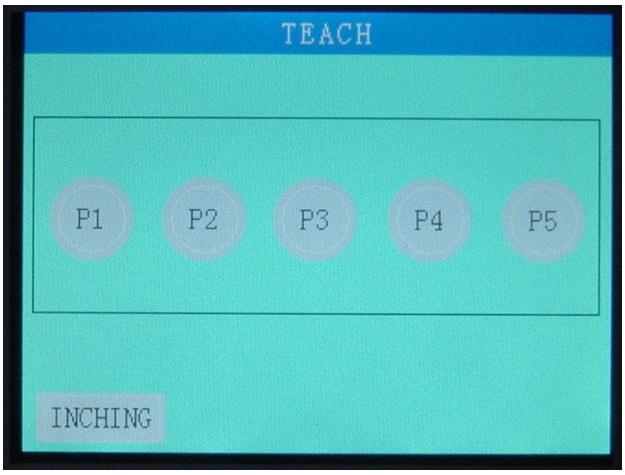
・POINT PARAMETER
ERC2 内部ポジションパラメータのうち、
位置、最高速度、加速度、減速度
を表示します。
各パラメータの表示部にタッチするとテンキーが表示され内容を変更することができます。
Write to RC ボタンで ERC2 に書き込みます。



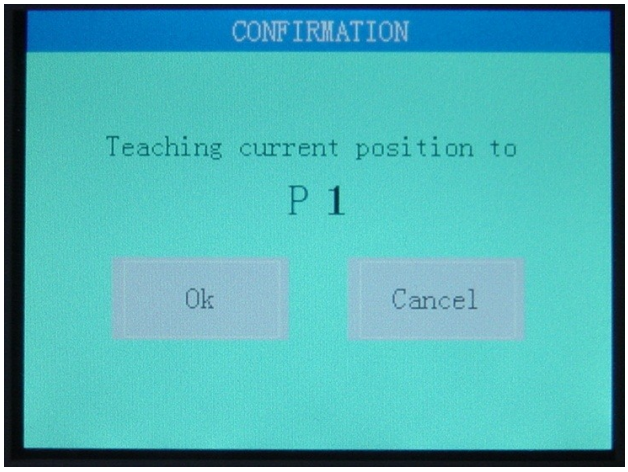
・MOVE
指定した ERC2 内部ポジションへ移動します。
このサンプルでは P1～P5 を使います。



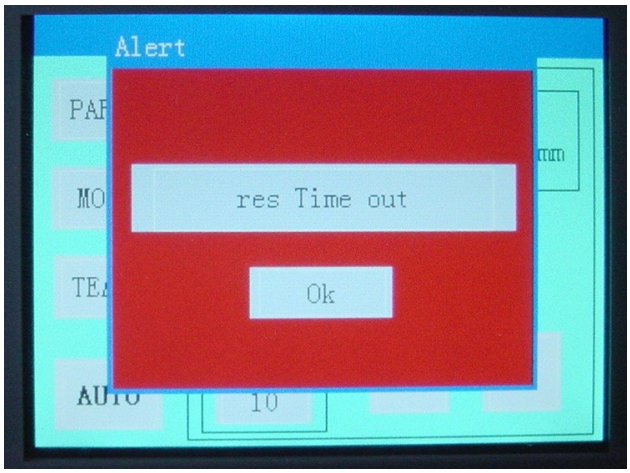
・TEACH
ERC2 内部ポジションを教示します。
ERC2 内部ポジションの位置は PARAM 画面でも直接変更できます。



・CONFIRMATION
ERC2 内部ポジションを教示するときの確認画面です。



・Alert
通信で入力 TIME OUT、LRC 不整合が発生した時のエラー表示です。



MPC サンプルプログラムで使用しているクエリ

名称	記号	サブルーチン
現在位置の読み取り	PNOW	*READ_PNOW
コントローラ状態信号の読み取り 1	DSS1	*CHECK_DSS1
原点復帰	HOME	*HOME
位置決め動作起動命令	CSTR	*MOVE_TEACH_POINT
ティーチモード指令	MOD	*TEACHING
ポジションデータ取り込み	TEAC	*TEACHING
ポジション No.指定移動	POSR	*TEACHING , *MOVE_TEACH_POINT
直値移動指令		*MOVE_ABS, *MOVE_REL
ポジションテーブルデータ書き込み		*WRITE_POINT_PARAM

・その他動作を確認したクエリ

アラームリセット	ALRS
一時停止	STP

MPC サンプルプログラム サブルーチンの説明【 Modbus ASCII 】

*COM_CNFG
通信ポートの設定と初期化を行います。

*LRC_CALC_SND
送信文字列と LRC 計算結果でクエリ文字列を作成し ERC2 に送信します。また、レスポンスの LRC を計算して受信文字列内の LRC 部と照合します。
レスポンスは文字列変数 RES$に格納し、親のサブルーチンで必要箇所を切り出して使用します。

*WAIT_MOVE_END
ERC2 の DSS1 レジスタを読み、原点復帰完了、移動完了を待ちます。

*HOME
原点復帰を行います。

*MOVE_TEACH_POINT
指定した ERC2 内部ポジションへ移動します。

*MOVE_REL
相対座標移動を行います。移動量指定は 0.01mm 単位です。

*MOVE_ABS
絶対座標移動を行います。指定座標は 0.01mm 単位です。

*READ_POINT_PARAM
ERC2 内部ポジションのパラメータを読み出し、MBK()に入れます。

*WRITE_POINT_PARAM
MBK()の ERC2 内部ポジションのパラメータを ERC2 へ書き込みます。

*READ_PNOW
現在位置を読み出します。

*CHECK_DSS1
DSS1 レジスタを読み出します。
サンプルプログラムでは先頭で原点復帰が完了しているか否かをチェックしています。

*AUTO
AUTO 動作画面です。ERC2 内部ポジション移動、絶対座標移動、相対座標移動を繰り返します。

*INCHING
タッチパネルのインチング移動画面に遷移し、設定した量だけ INCHING 移動を行います。

*MOVE

タッチパネルのポジション移動画面に遷移し、選択したポジションに移動します。

*TEACH
タッチパネルのティーチング画面に遷移し、選択したポジションを教示します。

*TEACH_CONFIRM
タッチパネルのティーチングの確認画面です。

*TEACHING
現在位置を ERC2 内部ポジションとして書き込みます。

*DISPLAY_CURRENT_POSITION
タッチパネルに現在位置を表示します。

*DISPLAY_AUTO_MESSEGE
タッチパネルに AUTO 動作中の状態を表示します。

*PARAM
タッチパネルに ERC2 ポジションの主要パラメータを表示します。変更も可能です。

MPC サンプルプログラム【 Modbus ASCII 】

```
  TIME 2000
  GOSUB *COM_CNFG                          /* initialize the communication ports
  S_MBK 0 0                                /* the opening screen

  FOR main_i=5000 TO 5240                  /* clear the MBK area for point parameter
    S_MBK 0 main_i
  NEXT

  S_MBK 100 100                            /* sets initial inching distance 1mm

  FOR main_i=1 TO 5                        /* read point parameters
    GOSUB *READ_POINT_PARAM main_i
  NEXT
  GOSUB *CHECK_DSS1                        /* DSS1 = device status register 1
  IF DSS1_STAT&&H10==0 THEN                /* DSS1_STAT is a result of *CHECK_DSS1
    msg$="Did not HOME"
  ELSE
    msg$=""
  END_IF
  S_MBK msg$ 220 12                        /* it's in the INCHING screen

*AUTO                                      /* AUTO process
  GOSUB *AUTO_PARAM_SET
  DO

    DO
      IF SW(71300)==1 THEN                 /* START button is on
        BREAK
      ELSE
        OFF 71302                          /* show the INCHING button
        msg$="<push START>"
        GOSUB *DISPLAY_AUTO_MESSEGE
      END_IF
      IF SW(71301)==1 THEN                 /* INCHING button is on
        GOSUB *INCHING
        GOSUB *AUTO_PARAM_SET
      END_IF
      SWAP
    LOOP

    ON 71302                               /* hide the INCHING button
    job_count=job_count+1
    S_MBK job_count 104

    IF  msg$=="<push START>" THEN          /* only first
      msg$="HOME"
      GOSUB *DISPLAY_AUTO_MESSEGE
      GOSUB *HOME
      S_MBK "" 220 12
    END_IF

    FOR pi=1 TO 5
      FORMAT "0"
```

```
      msg$="move to P"+STR$(pi)
      GOSUB *DISPLAY_AUTO_MESSEGE
      GOSUB *MOVE_TEACH_POINT pi
      TIME 100
    NEXT

    msg$="ABS move 0.00mm"
    GOSUB *DISPLAY_AUTO_MESSEGE
    GOSUB *MOVE_ABS 0 30000 90

    msg$="ABS move 100.00mm"
    GOSUB *DISPLAY_AUTO_MESSEGE
    GOSUB *MOVE_ABS 10000 30000 90

    TIME 200

    FOR pj=1 TO 2
      FOR pi=1 TO 3
        msg$="REL move -10.00mm"
        GOSUB *DISPLAY_AUTO_MESSEGE
        GOSUB *MOVE_REL -1000 20000 90
      NEXT
      TIME 200
    NEXT

    FOR pi=1 TO 7
      msg$="REL move -5.71mm"
      GOSUB *DISPLAY_AUTO_MESSEGE
      GOSUB *MOVE_REL -571 20000 90
      TIME 25
    NEXT

    TIME 500

  LOOP

*AUTO_PARAM_SET
  S_MBK 4 0
  OFF 71300                                   /* off the START button
  OFF 71302                                   /* show the INCHING button
  job_count=0
  S_MBK job_count 104
  RETURN

*INCHING
  S_MBK 1 0
  DO
    GOSUB *DISPLAY_CURRENT_POSITION

    in_stat=0
    WHILE in_stat==0
      in_stat=IN(71000~Wrd)
      SWAP
    WEND

    SELECT_CASE in_stat
      CASE &H01
        GOSUB *HOME
        S_MBK "" 220 12
      CASE &H02
        GOSUB *MOVE_REL MBK(100) 2000 30
      CASE &H04
        GOSUB *MOVE_REL MBK(100)*-1 2000 30
      CASE &H10
        GOSUB *MOVE                           /* the MOVE screen
      CASE &H20
        GOSUB *TEACH                          /* the TEACH screen
      CASE &H40
        RETURN                                /* return to AUTO
      CASE &H80
        GOSUB *PARAM                          /* the Point Parameters screen
      CASE_ELSE
        PRINT "in_stat?" in_stat
    END_SELECT
    WAIT IN(71000)==0
    S_MBK 1 0
    TIME 200

  LOOP
  RETURN

*COM_CNFG                                     /* defines communication port
  /*RS485_PORT=4
  /*MONI_PORT=2
```

```
  RS485_PORT=1
  MONI_PORT=0
  CNFG# RS485_PORT RS485 "38400b8pns1NONE"      /* initialize RS485 port
  TIME 500                                      /* delay need
  IF MONI_PORT<>0 THEN
    CNFG# MONI_PORT "38400b8pns1NONE"             /* monitor port for debugging
  END_IF
  MEWNET 38400 2                                /* for a touch panel
  RETURN

*PARAM
  S_MBK 5 0

  DO
    FOR main_i=1 TO 5
      GOSUB *READ_POINT_PARAM main_i
    NEXT
    WAIT SW(71400)|SW(71401)
    IF SW(71400)==1 THEN
      S_MBK "Writing..." 320 12
      ON 71402
      FOR main_i=1 TO 5
        GOSUB *WRITE_POINT_PARAM main_i
      NEXT
      TIME 1000
      WAIT SW(71400)==0
      OFF 71402
    END_IF
    IF SW(71401)==1 THEN
      BREAK
    END_IF

  LOOP
  RETURN


*DISPLAY_AUTO_MESSEGE                          /* display a message
  S_MBK msg$ 300 20
  RETURN

*MOVE                                          /* MOVE button process
  S_MBK 2 0
  S_MBK "" 340 20
  S_MBK 1 101

  DO
    WAIT IN(71100~Wrd)<>0
    SELECT_CASE IN(71100~Wrd)
      CASE &H01
        GOSUB *MOVE_ABS MBK(5042~Lng) 2000 30
        S_MBK "P1" 340 20
      CASE &H02
        GOSUB *MOVE_ABS MBK(5082~Lng) 2000 30
        S_MBK "P2" 340 20
      CASE &H04
        GOSUB *MOVE_ABS MBK(5122~Lng) 2000 30
        S_MBK "P3" 340 20
      CASE &H08
        GOSUB *MOVE_ABS MBK(5162~Lng) 2000 30
        S_MBK "P4" 340 20
      CASE &H10
        GOSUB *MOVE_ABS MBK(5202~Lng) 2000 30
        S_MBK "P5" 340 20
      CASE &H100
        BREAK
      CASE_ELSE
        PRINT "?"
    END_SELECT
    WAIT IN(71100~Wrd)==0

  LOOP
  RETURN

*TEACH                                         /* TEACH button process
  S_MBK 2 0
  S_MBK 2 101
  S_MBK "" 340 20

  WAIT IN(71100~Wrd)<>0
  SELECT_CASE IN(71100~Wrd)
    CASE &H01
      GOSUB *TEACH_CONFIRM 1
    CASE &H02
      GOSUB *TEACH_CONFIRM 2
```

```
    CASE &H04
      GOSUB *TEACH_CONFIRM 3
    CASE &H08
      GOSUB *TEACH_CONFIRM 4
    CASE &H10
      GOSUB *TEACH_CONFIRM 5
    CASE &H100
      BREAK
    CASE_ELSE
      PRINT "?"
  END_SELECT
  WAIT IN(71100~Wrd)==0

  RETURN

*TEACH_CONFIRM                          /* a confirming window
_VAR teaching_point
  S_MBK 3 0
  S_MBK teaching_point 102
  WAIT SW(71200)|SW(71201)
  IF SW(71200) THEN
    GOSUB *TEACHING teaching_point
    GOSUB *READ_POINT_PARAM teaching_point
  END_IF
  RETURN

*DISPLAY_CURRENT_POSITION                /* display a current position
  GOSUB *READ_PNOW
  S_MBK NOWP 200~Lng
  RETURN

'=====================================
' Cheking DDS1 Register
'=====================================

*CHECK_DSS1
  CMD$="010390050001"                   /* query DSS1
  GOSUB *LRC_CALC_SND
  /* RES$ format
  /*   :010302308842                    /* After Power ON
  /*   :010302309832                    /* After HOME
  /*   : start
  /*    01 slave adr
  /*      03 function code
  /*        02 data count
  /*          3088 data
  /*              4A LRC
  ptr_=RES$+7
  DSS1_STAT=HEX(PTR$(4))
  /*PRX DSS1_STAT
  RETURN

'=====================================
' Teaching Current Position
'=====================================
*TEACHING
  _VAR teach_point_no

  CMD$="01050414FF00"                   /* set query MOD
  GOSUB *LRC_CALC_SND

  FORMAT "00"
  CMD$="01060D0300"+HEX$(teach_point_no)  /* query POSR
  GOSUB *LRC_CALC_SND

  CMD$="01050415FF00"                   /* set query TEACH
  GOSUB *LRC_CALC_SND
  TIME 50
  CMD$="010504150000"                   /* reset query TEACH
  GOSUB *LRC_CALC_SND

  CMD$="010504140000"                   /* reset query MOD
  GOSUB *LRC_CALC_SND
  RETURN

'=====================================
' Read Current Position
'=====================================
*READ_PNOW
  CMD$="010390000002"                   /* query PNOW
  GOSUB *LRC_CALC_SND
  /* RES$ format
  /*  ":010304000013885D<cr><lf>"
  /*   : start
```

```
  /*    01 slave adr
  /*      03 function code
  /*        04 data byte count
  /*          0000 data high
  /*              1388 data low 0x1388= 5000(*0.01mm)
  /*                  5D LRC

  ptr_=RES$+7
  NOWP=HEX(PTR$(8))  /* copy data part
  /*PRINT "Current Position " NOWP
  RETURN

'=====================================
' Write Point Datas
'=====================================
*WRITE_POINT_PARAM
_VAR point_no
/* Command Format
/* 011010C0 000F1E000027100000000A00004E200000177000000FA00001001E000000000000
/* 01 slave address
/*   10 function code
/*     10C0 start adress 10C0 = &H1000 + &HC0 , &HC0=192 = 16*12
/*          000F register count
/*              1E byte count
/*                00002710 PCMD
/*                        0000000A INP
/*                                00004E20 VCMD
/*                                        00001770 ZNMP
/*                                                00000FA0 ZNLP
/*                                                        0001 ACMD
/*                                                            001E DCMD
/*                                                                0000 PPOW
/*                                                                    0000 LPOW
/*                                                                        0000 CTLF
  po=point_no*40+5000
  FORMAT "0000"
  data_start$=HEX$(MBK(po~Lng)*16+&H1000)
  FORMAT "00000000"
  data_pcmd$=HEX$(MBK(po+2~Lng))
  data_inp$=HEX$(MBK(po+4~Lng))
  data_vcmd$=HEX$(MBK(po+6~Lng))
  data_znmp$=HEX$(MBK(po+8~Lng))
  data_znlp$=HEX$(MBK(po+10~Lng))
  FORMAT "0000"
  data_acmd$=HEX$(MBK(po+12~Lng))
  data_dcmd$=HEX$(MBK(po+14~Lng))
  data_ppow$=HEX$(MBK(po+16~Lng))
  data_lpow$=HEX$(MBK(po+18~Lng))
  data_ctlf$=HEX$(MBK(po+20~Lng))

  CMD$="0110"                                   /* slave address function code
  CMD$=CMD$+data_start$                         /* start adress
  CMD$=CMD$+"000F"                              /* register count
  CMD$=CMD$+"1E"                                /* byte count
  CMD$=CMD$+data_pcmd$                          /* PCMD
  CMD$=CMD$+data_inp$                           /* INP
  CMD$=CMD$+data_vcmd$                          /* VCMD
  CMD$=CMD$+data_znmp$                          /* ZNMP
  CMD$=CMD$+data_znlp$                          /* ZNLP
  CMD$=CMD$+data_acmd$                          /* ACMD
  CMD$=CMD$+data_dcmd$                          /* DCMD
  CMD$=CMD$+data_ppow$                          /* PPOW
  CMD$=CMD$+data_lpow$                          /* LPOW
  CMD$=CMD$+data_ctlf$                          /* CTLF

  /*PR CMD$
  GOSUB *LRC_CALC_SND
  RETURN

'=====================================
' Read Point Datas
'=====================================
*READ_POINT_PARAM
_VAR point_no
/* RES$ format
/* 0          1          2          3          4          5          6          7
/* 01234 56 78901234 56789012 34567890 12345678 90123456 7890 1234 5678 9012 3456 789 01
/* :0103 1E 00001F41 0000000A 00007530 00000000 00000000 005A 005A 0000 0000 0000 A39 4E
/*       BC PCMD     INP      VCMD     ZNMP     ZNLP     ACMD DCMD PPOW LPOW CTLF ?   LRC
/*    BC=ByteCount

  CMD$="0103"
  point_no1=point_no*16+&H1000
```

```
  FORMAT "0000"
  CMD$=CMD$+HEX$(point_no1)+"000F"

  GOSUB *LRC_CALC_SND
  ptr_=RES$+7
  data_pcmd=HEX(PTR$(8))                          /* destination
  ptr_=RES$+15
  data_inp=HEX(PTR$(8))                           /* in-position width
  ptr_=RES$+23
  data_vcmd=HEX(PTR$(8))                          /* speed
  ptr_=RES$+31
  data_znmp=HEX(PTR$(8))                          /* zone +
  ptr_=RES$+39
  data_znlp=HEX(PTR$(8))                          /* zone -
  ptr_=RES$+47
  data_acmd=HEX(PTR$(4))                          /* acceleration
  ptr_=RES$+51
  data_dcmd=HEX(PTR$(4))                          /* deceleration
  ptr_=RES$+55
  data_ppow=HEX(PTR$(4))                          /* pushing current limit
  ptr_=RES$+59
  data_lpow=HEX(PTR$(4))                          /* load current  threshold
  ptr_=RES$+63
  data_ctlf=HEX(PTR$(4))                          /* control flag

  po=point_no*40+5000
  FILL MBK(po) 40 0                               /* data clear
  MBK(po~Lng)=point_no
  MBK(po+2~Lng)=data_pcmd
  MBK(po+4~Lng)=data_inp
  MBK(po+6~Lng)=data_vcmd
  MBK(po+8~Lng)=data_znmp
  MBK(po+10~Lng)=data_znlp
  MBK(po+12~Lng)=data_acmd
  MBK(po+14~Lng)=data_dcmd
  MBK(po+16~Lng)=data_ppow
  MBK(po+18~Lng)=data_lpow
  MBK(po+20~Lng)=data_ctlf
  /*PRINT "point " point_no data_pcmd data_inp data_vcmd data_acmd data_dcmd
  RETURN

'=====================================
' Absolute Move
'=====================================
*MOVE_ABS
_VAR dest speed accel
/* Command Format
/* "011099000009120000138800000000A00002710001E00000000"
/*  01 slave address
/*    10 function code
/*      9900 start address
/*          0009 register count
/*              12 byte count
/*                0000 destination point high
/*                    1388 destination point low 50mm*100=5000 = 0x1388
/*                        0000 in-position width high
/*                            000A in-position width low
/*                                0000 speed high
/*                                    2710 speed low 100mm/sec*100=10000 = 0x2710
/*                                        001E acceleration/deceleration  0.3G*100=30 = 0x1E
/*                                            0000 press
/*                                                0000 control flag

  CMD$="01109900000912"
  FORMAT "00000000"
  CMD$=CMD$+HEX$(dest)                            /* destination = 2 word
  CMD$=CMD$+"0000000A"                            /* in-position width = 2 word
  CMD$=CMD$+HEX$(speed)                           /* speed = 2 word
  FORMAT "0000"
  CMD$=CMD$+HEX$(accel)                           /* acceleration/deceleration = word
  CMD$=CMD$+"00000000"                            /* flag

  GOSUB *LRC_CALC_SND
  TIME 5                                          /* Is this necessity ?
  GOSUB *WAIT_MOVE_END &H8                        /* &H8=PEND status
  RETURN

'=====================================
' Relative Move
'=====================================
*MOVE_REL
_VAR dest speed accel
/* Command Format
/* "011099000009120000138800000000A00002710001E00000008"
```

```
/*  01 slave address
/*    10 function code
/*      9900 start address
/*          0009 register count
/*              12 byte count
/*                0000 destination point high
/*                    1388 destination point low 50mm*100=5000 = 0x1388
/*                        0000 in-position width high
/*                            000A in-position width low
/*                                0000 speed high
/*                                    2710 speed low 100mm/sec*100=10000 = 0x2710
/*                                        001E acceleration/deceleration  0.3G*100=30 = 0x1E
/*                                            0000 press
/*                                                0008 control flag

  CMD$="01109900000912"
  FORMAT "00000000"
  CMD$=CMD$+HEX$(dest)                          /* destination = 2 word
  CMD$=CMD$+"0000000A"                          /* in-position width = 2 word
  CMD$=CMD$+HEX$(speed)                         /* speed = 2 word
  FORMAT "0000"
  CMD$=CMD$+HEX$(accel)                         /* acceleration/deceleration = word
  CMD$=CMD$+"00000008"                          /* flag

  GOSUB *LRC_CALC_SND
  TIME 5                                        /* Is this necessity ?
  GOSUB *WAIT_MOVE_END &H08                     /* &H8=PEND status
  RETURN

'=====================================
' Move to a Teaching Point
'=====================================
*MOVE_TEACH_POINT
_VAR teach_point_no

  FORMAT "00"
  CMD$="01060D0300"+HEX$(teach_point_no)        /* query POSR
  GOSUB *LRC_CALC_SND

  CMD$="0105040C0000"                           /* reset query CSTR
  GOSUB *LRC_CALC_SND
  CMD$="0105040CFF00"                           /* set query CSTR
  GOSUB *LRC_CALC_SND
  /* if CSTR=0xFF then doesn't activate PEND status
  CMD$="0105040C0000"                           /* reset query CSTR
  GOSUB *LRC_CALC_SND
  GOSUB *WAIT_MOVE_END &H08                     /* &H8=PEND status
  RETURN

'=====================================
' Go to the origin
'=====================================
*HOME
  CMD$="0105040B0000"                           /* reset query HOME
  GOSUB *LRC_CALC_SND
  CMD$="0105040BFF00"                           /* set query HOME
  GOSUB *LRC_CALC_SND
  CMD$="0105040B0000"                           /* reset query HOME
  GOSUB *LRC_CALC_SND
  GOSUB *WAIT_MOVE_END &H10                     /* &H10=HEND status
  RETURN

'=====================================
' reading the DSS1 and checking status
'=====================================
*WAIT_MOVE_END
_VAR chek_bit
/* chek_bit &H0010=HEND, &H0008=PEND
  DO
    CMD$="010390050001"                         /* query DSS1
    GOSUB *LRC_CALC_SND
    /* RES$ format
    /*  ":01030230804A"
    /*   : start
    /*    01 slave adr
    /*      03 function code
    /*        02 data count
    /*          3080 data
    /*              4A LRC
    ptr_=RES$+7
    dss1_stat=HEX(PTR$(4))                      /* copy data part
    IF dss1_stat&chek_bit<>0 THEN
      BREAK
    END_IF
```

```
  LOOP
  RETURN

'=====================================
' LCR calculate
'=====================================
*LRC_CALC_SND
  send_cmd$=CMD$
  calc_ptr=send_cmd$                              /* get pointer of send_cmd$
  calc_sum=0
  calc_len=LEN(send_cmd$)
  FOR calc_i=1 TO calc_len STEP 2
    ptr_=calc_ptr                                 /* set pointer for PTR$()
    calc_sum=calc_sum+HEX(PTR$(2))                /* get hex code of 2 characters
    calc_ptr=calc_ptr+2
  NEXT
  calc_sum=&H100-(calc_sum&&HFF)
  FORMAT "00"
  calc_sum$=HEX$(calc_sum)

  send_cmd$=":"+send_cmd$+calc_sum$+"¥r¥n"                /* make send string
  PRINT# RS485_PORT  send_cmd$
  IF MONI_PORT<>0 THEN
    PRINT# MONI_PORT  send_cmd$
    PRINT# MONI_PORT ">"
  END_IF

  RES$=""
  DO
    INPUT# RS485_PORT CHR_C|1 TMOUT|3 calc_buf$
    IF rse_<>0 THEN
      ALERT_MSG$="res Time out"
      GOSUB *DISPLAY_ALERT
      GOTO *LRC_CALC_SND                          /* retry
    END_IF

    IF MONI_PORT<>0 THEN
      PRINT# MONI_PORT STR_LEN|1 calc_buf$        /* monitor output for debug
    END_IF
    RES$=RES$+calc_buf$
    IF calc_buf$=CHR$(&HA) THEN
      BREAK
    END_IF
    SWAP
  LOOP
  ptr_=RES$+LEN(RES$)-4
  res_lrc=HEX(PTR$(2))                            /* get LRC in a response

  calc_ptr=RES$+1                                 /* pointer of next to ":"
  calc_sum=0
  calc_len=LEN(RES$)-5                            /* except for  ":" LRC "¥r¥n"
  FOR calc_i=1 TO calc_len STEP 2
    SWAP
    ptr_=calc_ptr                                 /* set pointer for PTR$()
    calc_sum=calc_sum+HEX(PTR$(2))                /* get hex code of 2 characters
    calc_ptr=calc_ptr+2
  NEXT
  calc_sum=&H100-(calc_sum&&HFF)
  calc_sum=calc_sum&&HFF

  /* PR RES$ HEX$(res_lrc) HEX$(calc_sum)

  IF res_lrc<>calc_sum THEN
    PRINT "res LRC error" res_lrc calc_sum
    FORMAT "00"
    ALERT_MSG$="res LRC err "+HEX$(res_lrc)+" "+HEX$(calc_sum)
    GOSUB *DISPLAY_ALERT
    GOTO *LRC_CALC_SND                            /* retry
  END_IF
  TIME 5                                          /* need (delay time between response and query)
RETURN

*DISPLAY_ALERT
  previous_screen=MBK(0)                          /* previous screen no.
  S_MBK ALERT_MSG$ 360 20                         /* display alert message
  OFF 71500
  S_MBK 7 0                                       /* open the alert window
  WAIT SW(71500)==1                               /* notification of the window close
  OFF 71500
  S_MBK previous_screen 0
  RETURN
```

MPC サンプルプログラム サブルーチンの説明 【 Modbus RTU 】

(Modbus ASII との主な相違箇所)

*RTU_QUERY_SND_RCV
　　コマンド文字列と CRC16 計算結果でクエリ文字列を組み立てて ERC2 に送信します。また、レスポンスの CRC16 計算結果と受信文字列内の CRC16 データ部を照合します。
　　コマンド文字列、レスポンス文字列は配列変数に入れて CRC16 を計算します。
　　レスポンスは文字列変数 RES$に格納し、親のサブルーチンで必要箇所を切り出して使用します。

*RTU_CRC_CALC
　　CRC16 方式で送受信文字列のチェックコードを生成します。

・使用しているクエリは Modbus ASCII と同じです。

MPC サンプルプログラム【 Modbus RTU 】

```
  DIM QUERY(128)                          /* using QUERY(1)~

  TIME 2000
  GOSUB *COM_CNFG                         /* initialize the communication ports
  S_MBK 0 0                               /* the opening screen

  FOR main_i=5000 TO 5240                 /* clear the MBK area for point parameter
    S_MBK 0 main_i
  NEXT

  S_MBK 100 100                           /* sets initial inching distance 1mm

  FOR main_i=1 TO 5                       /* read point parameters
    GOSUB *READ_POINT_PARAM main_i
  NEXT
  GOSUB *CHECK_DSS1                       /* DSS1 = device status register 1
  IF DSS1_STAT&&H10==0 THEN               /* DSS1_STAT is a result of *CHECK_DSS1
    MSG$="Did not HOME"
  ELSE
    MSG$=""
  END_IF
  S_MBK MSG$ 220 12                       /* it's in the INCHING screen

*AUTO                                     /* AUTO process
  GOSUB *AUTO_PARAM_SET
  MSG$="Modbus RTU"
  GOSUB *DISPLAY_AUTO_MESSEGE
  TIME 2000

  DO

    DO
      IF SW(71300)==1 THEN                /* START button is on
        BREAK
      ELSE
        OFF 71302                         /* show the INCHING button
        MSG$="<push START>"
        GOSUB *DISPLAY_AUTO_MESSEGE
      END_IF
      IF SW(71301)==1 THEN                /* INCHING button is on
        GOSUB *INCHING
        GOSUB *AUTO_PARAM_SET
      END_IF
      SWAP
    LOOP

    ON 71302                              /* hide the INCHING button
    job_count=job_count+1
    S_MBK job_count 104

    IF  MSG$=="<push START>" THEN         /* only first
      MSG$="HOME"
      GOSUB *DISPLAY_AUTO_MESSEGE
      GOSUB *HOME
      S_MBK "" 220 12
    END_IF

    FOR pi=1 TO 5
      FORMAT "0"
```

```
      MSG$="move to P"+STR$(pi)
      GOSUB *DISPLAY_AUTO_MESSEGE
      GOSUB *MOVE_TEACH_POINT pi
      TIME 100
    NEXT

    MSG$="ABS move 0.00mm"
    GOSUB *DISPLAY_AUTO_MESSEGE
    GOSUB *MOVE_ABS 0 30000 90

    MSG$="ABS move 100.00mm"
    GOSUB *DISPLAY_AUTO_MESSEGE
    GOSUB *MOVE_ABS 10000 30000 90

    TIME 200

    FOR pj=1 TO 2
      FOR pi=1 TO 3
        MSG$="REL move -10.00mm"
        GOSUB *DISPLAY_AUTO_MESSEGE
        GOSUB *MOVE_REL -1000 20000 90
      NEXT
      TIME 200
    NEXT

    FOR pi=1 TO 6
      MSG$="REL move -5.71mm"
      GOSUB *DISPLAY_AUTO_MESSEGE
      GOSUB *MOVE_REL -571 10000 90
    NEXT
    TIME 25
    GOSUB *MOVE_ABS 0 10000 90

    TIME 500
  LOOP

*AUTO_PARAM_SET
  S_MBK 4 0
  OFF 71300                                   /* off the START button
  OFF 71302                                   /* show the INCHING button
  job_count=0
  S_MBK job_count 104
  RETURN

*INCHING
  S_MBK 1 0
  DO
    GOSUB *DISPLAY_CURRENT_POSITION

    in_stat=0
    WHILE in_stat==0
      in_stat=IN(71000~Wrd)
      SWAP
    WEND

    SELECT_CASE in_stat
      CASE &H01
        GOSUB *HOME
        S_MBK "" 220 12
      CASE &H02
        GOSUB *MOVE_REL MBK(100) 2000 30
      CASE &H04
        GOSUB *MOVE_REL MBK(100)*-1 2000 30
      CASE &H10
        GOSUB *MOVE                           /* the MOVE screen
      CASE &H20
        GOSUB *TEACH                          /* the TEACH screen
      CASE &H40
        RETURN                                /* return to AUTO
      CASE &H80
        GOSUB *PARAM                          /* the Point Parameters screen
      CASE_ELSE
        PRINT "in_stat?" in_stat
    END_SELECT
    WAIT IN(71000)==0
    S_MBK 1 0
    TIME 200

  LOOP

  RETURN

*COM_CNFG                                     /* defines communication port
  /*RS485_PORT=4
```

```
  /*MONI_PORT=2
  RS485_PORT=1
  MONI_PORT=0
  MEWNET 38400 2                                  /* define a touch panel port

  CNFG# RS485_PORT RS485 "38400b8pns1NONE"        /* initialize RS485 port
  TIME 500                                        /* delay need
  IF MONI_PORT<>0 THEN
    CNFG# MONI_PORT "38400b8pns1NONE"             /* monitor port for debugging
  END_IF
  RETURN

*PARAM
  S_MBK 5 0
  DO
    FOR main_i=1 TO 5
      GOSUB *READ_POINT_PARAM main_i
    NEXT
    WAIT SW(71400)|SW(71401)
    IF SW(71400)==1 THEN
      S_MBK "Writing..." 320 12
      ON 71402
      FOR main_i=1 TO 5
        GOSUB *WRITE_POINT_PARAM main_i
      NEXT
      TIME 1000
      WAIT SW(71400)==0
      OFF 71402
    END_IF
    IF SW(71401)==1 THEN
      BREAK
    END_IF
  LOOP
  RETURN

*DISPLAY_AUTO_MESSEGE                            /* display a message
  S_MBK MSG$ 300 20
  RETURN

*MOVE                                            /* MOVE button process
  S_MBK 2 0
  S_MBK "" 340 20
  S_MBK 1 101
  DO
    WAIT IN(71100~Wrd)<>0
    SELECT_CASE IN(71100~Wrd)
      CASE &H01
        GOSUB *MOVE_ABS MBK(5042~Lng) 2000 30
        S_MBK "P1" 340 20
      CASE &H02
        GOSUB *MOVE_ABS MBK(5082~Lng) 2000 30
        S_MBK "P2" 340 20
      CASE &H04
        GOSUB *MOVE_ABS MBK(5122~Lng) 2000 30
        S_MBK "P3" 340 20
      CASE &H08
        GOSUB *MOVE_ABS MBK(5162~Lng) 2000 30
        S_MBK "P4" 340 20
      CASE &H10
        GOSUB *MOVE_ABS MBK(5202~Lng) 2000 30
        S_MBK "P5" 340 20
      CASE &H100
        BREAK
      CASE_ELSE
        PRINT "?"
    END_SELECT
    WAIT IN(71100~Wrd)==0
  LOOP
  RETURN

*TEACH                                           /* TEACH button process
  S_MBK 2 0
  S_MBK 2 101
  S_MBK "" 340 20
  WAIT IN(71100~Wrd)<>0
  SELECT_CASE IN(71100~Wrd)
    CASE &H01
      GOSUB *TEACH_CONFIRM 1
    CASE &H02
      GOSUB *TEACH_CONFIRM 2
    CASE &H04
      GOSUB *TEACH_CONFIRM 3
    CASE &H08
      GOSUB *TEACH_CONFIRM 4
```

```
    CASE &H10
      GOSUB *TEACH_CONFIRM 5
    CASE &H100
      BREAK
    CASE_ELSE
      PRINT "?"
  END_SELECT
  WAIT IN(71100~Wrd)==0
  RETURN

*TEACH_CONFIRM                          /* confirming window
_VAR teaching_point
  S_MBK 3 0
  S_MBK teaching_point 102
  WAIT SW(71200)|SW(71201)
  IF SW(71200) THEN
    GOSUB *TEACHING teaching_point
    GOSUB *READ_POINT_PARAM teaching_point
  END_IF
  RETURN

*DISPLAY_CURRENT_POSITION                /* display a current position
  GOSUB *READ_PNOW
  S_MBK NOWP 200~Lng
  RETURN

'=====================================
' Cheking DDS1 Register
'=====================================
*CHECK_DSS1
  CMD$="010390050001"                   /* query DSS1
  GOSUB *RTU_QUERY_SND_RCV 7
  /* RES$ format
  /* ptr_+01234567
  /*      0103023088AC22                 /* After Power-on
  /*      01 slave adr
  /*        03 function code
  /*          02 data count
  /*            3088 data
  /*                AC22 CRC16
  ptr_=RES$+6
  DSS1_STAT=HEX(PTR$(4))
  /*PRX DSS1_STAT
  RETURN

'=====================================
' Teaching Current Position
'=====================================
*TEACHING
  _VAR teach_point_no

  CMD$="01050414FF00"                   /* set query MOD
  GOSUB *RTU_QUERY_SND_RCV 8
  FORMAT "00"
  CMD$="01060D0300"+HEX$(teach_point_no)  /* query POSR
  GOSUB *RTU_QUERY_SND_RCV 8
  CMD$="01050415FF00"                   /* set query TEACH
  GOSUB *RTU_QUERY_SND_RCV 8
  TIME 50
  CMD$="010504150000"                   /* reset query TEACH
  GOSUB *RTU_QUERY_SND_RCV 8
  CMD$="010504140000"                   /* reset query MOD
  GOSUB *RTU_QUERY_SND_RCV 8
  RETURN

'=====================================
' Read Current Position
'=====================================
*READ_PNOW
  CMD$="010390000002"                   /* query PNOW
  GOSUB *RTU_QUERY_SND_RCV 9
  /* RES$ format
  /* ptr_+012345678901234567
  /*      01030400001388F765
  /*      01 slave adr
  /*        03 function code
  /*          04 data byte count
  /*            00001388 data 0x00001388= 5000(*0.01mm)
  /*                    F765 CRC
  ptr_=RES$+6
  NOWP=HEX(PTR$(8))  /* copy data part
  /*PRINT "Current Position " NOWP
  RETURN
```

```
'=====================================
' Write Point Datas
'=====================================
*WRITE_POINT_PARAM
_VAR point_no
/* Command Format
/* 011010C0 000F1E000027100000000A00004E200000177000000FA00001001E000000000000
/* 01 slave address
/*   10 function code
/*     10C0 start adress 10C0 = &H1000 + &HC0 , &HC0=192 = 16*12
/*          000F register count
/*              1E byte count
/*                00002710 PCMD
/*                         0000000A INP
/*                                  00004E20 VCMD
/*                                           00001770 ZNMP
/*                                                    00000FA0 ZNLP
/*                                                             0001 ACMD
/*                                                                  001E DCMD
/*                                                                       0000 PPOW
/*                                                                            0000 LPOW
/*                                                                                 0000 CTLF
  po=point_no*40+5000
  FORMAT "0000"
  data_start$=HEX$(MBK(po~Lng)*16+&H1000)
  FORMAT "00000000"
  data_pcmd$=HEX$(MBK(po+2~Lng))
  data_inp$=HEX$(MBK(po+4~Lng))
  data_vcmd$=HEX$(MBK(po+6~Lng))
  data_znmp$=HEX$(MBK(po+8~Lng))
  data_znlp$=HEX$(MBK(po+10~Lng))
  FORMAT "0000"
  data_acmd$=HEX$(MBK(po+12~Lng))
  data_dcmd$=HEX$(MBK(po+14~Lng))
  data_ppow$=HEX$(MBK(po+16~Lng))
  data_lpow$=HEX$(MBK(po+18~Lng))
  data_ctlf$=HEX$(MBK(po+20~Lng))

  CMD$="0110"                                   /* slave address function code
  CMD$=CMD$+data_start$                         /* start adress
  CMD$=CMD$+"000F"                              /* register count
  CMD$=CMD$+"1E"                                /* byte count
  CMD$=CMD$+data_pcmd$                          /* PCMD
  CMD$=CMD$+data_inp$                           /* INP
  CMD$=CMD$+data_vcmd$                          /* VCMD
  CMD$=CMD$+data_znmp$                          /* ZNMP
  CMD$=CMD$+data_znlp$                          /* ZNLP
  CMD$=CMD$+data_acmd$                          /* ACMD
  CMD$=CMD$+data_dcmd$                          /* DCMD
  CMD$=CMD$+data_ppow$                          /* PPOW
  CMD$=CMD$+data_lpow$                          /* LPOW
  CMD$=CMD$+data_ctlf$                          /* CTLF
  /*PR CMD$
  GOSUB *RTU_QUERY_SND_RCV 8
  RETURN

'=====================================
' Read Point Datas
'=====================================
*READ_POINT_PARAM
_VAR point_no
/* RES$ format
/*      0          1          2          3          4          5          6          7
/* ptr_+0123 45 67890123 45678901 23456789 01234567 89012345 6789 0123 4567 8901 2345 6789 01
/*      0103 1E 000007D0 0000000A 000007D0 00000000 00000000 005A 005A 0000 0000 0000 E309
/*           BC PCMD     INP      VCMD     ZNMP     ZNLP     ACMD DCMD PPOW LPOW CTLF CRC
/*           BC=ByteCount

  CMD$="0103"
  point_no1=point_no*16+&H1000
  FORMAT "0000"
  CMD$=CMD$+HEX$(point_no1)+"000F"
  GOSUB *RTU_QUERY_SND_RCV 35
  ptr_=RES$+6
  data_pcmd=HEX(PTR$(8))                        /* destination
  ptr_=RES$+14
  data_inp=HEX(PTR$(8))                         /* in-position width
  ptr_=RES$+22
  data_vcmd=HEX(PTR$(8))                        /* speed
  ptr_=RES$+30
  data_znmp=HEX(PTR$(8))                        /* zone +
  ptr_=RES$+38
  data_znlp=HEX(PTR$(8))                        /* zone -
  ptr_=RES$+46
```

```
  data_acmd=HEX(PTR$(4))                        /* acceleration
  ptr_=RES$+50
  data_dcmd=HEX(PTR$(4))                        /* deceleration
  ptr_=RES$+54
  data_ppow=HEX(PTR$(4))                        /* pushing current limit
  ptr_=RES$+58
  data_lpow=HEX(PTR$(4))                        /* load current  threshold
  ptr_=RES$+62
  data_ctlf=HEX(PTR$(4))                        /* control flag

  po=point_no*40+5000
  FILL MBK(po) 40 0                             /* data clear
  MBK(po~Lng)=point_no
  MBK(po+2~Lng)=data_pcmd
  MBK(po+4~Lng)=data_inp
  MBK(po+6~Lng)=data_vcmd
  MBK(po+8~Lng)=data_znmp
  MBK(po+10~Lng)=data_znlp
  MBK(po+12~Lng)=data_acmd
  MBK(po+14~Lng)=data_dcmd
  MBK(po+16~Lng)=data_ppow
  MBK(po+18~Lng)=data_lpow
  MBK(po+20~Lng)=data_ctlf
  /*PRINT "point " point_no data_pcmd data_inp data_vcmd data_acmd data_dcmd
  RETURN

'=====================================
' Absolute Move
'=====================================
*MOVE_ABS
_VAR dest speed accel
/* Command Format
/* "01109900000912000013880000000A00002710001E00000000"
/*  01 slave address
/*    10 function code
/*      9900 start address
/*          0009 register count
/*              12 byte count
/*                0000 destination point high
/*                    1388 destination point low 50mm*100=5000 = 0x1388
/*                        0000 in-position width high
/*                            000A in-position width low
/*                                0000 speed high
/*                                    2710 speed low 100mm/sec*100=10000 = 0x2710
/*                                        001E acceleration/deceleration  0.3G*100=30 = 0x1E
/*                                            0000 press
/*                                                0000 control flag

  CMD$="01109900000912"
  FORMAT "00000000"
  CMD$=CMD$+HEX$(dest)                          /* destination = 2 word
  CMD$=CMD$+"0000000A"                          /* in-position width = 2 word
  CMD$=CMD$+HEX$(speed)                         /* speed = 2 word
  FORMAT "0000"
  CMD$=CMD$+HEX$(accel)                         /* acceleration/deceleration = word
  CMD$=CMD$+"00000000"                          /* flag
  GOSUB *RTU_QUERY_SND_RCV 8
  TIME 5                                        /* Is this necessity ?
  GOSUB *WAIT_MOVE_END &H8                      /* &H8=PEND status
  RETURN

'=====================================
' Relative Move
'=====================================
*MOVE_REL
_VAR dest speed accel
/* Command Format
/* "01109900000912000013880000000A00002710001E00000008"
/*  01 slave address
/*    10 function code
/*      9900 start address
/*          0009 register count
/*              12 byte count
/*                0000 destination point high
/*                    1388 destination point low 50mm*100=5000 = 0x1388
/*                        0000 in-position width high
/*                            000A in-position width low
/*                                0000 speed high
/*                                    2710 speed low 100mm/sec*100=10000 = 0x2710
/*                                        001E acceleration/deceleration  0.3G*100=30 = 0x1E
/*                                            0000 press
/*                                                0008 control flag
  CMD$="01109900000912"
  FORMAT "00000000"
```

```
  CMD$=CMD$+HEX$(dest)                          /* destination = 2 word
  CMD$=CMD$+"0000000A"                          /* in-position width = 2 word
  CMD$=CMD$+HEX$(speed)                         /* speed = 2 word
  FORMAT "0000"
  CMD$=CMD$+HEX$(accel)                         /* acceleration/deceleration = word
  CMD$=CMD$+"00000008"                          /* flag
  GOSUB *RTU_QUERY_SND_RCV 8
  TIME 5                                        /* Is this necessity ?
  GOSUB *WAIT_MOVE_END &H08                     /* &H8=PEND status
  RETURN

'=====================================
' Move to a Teaching Point
'=====================================
*MOVE_TEACH_POINT
_VAR teach_point_no
  FORMAT "00"
  CMD$="01060D0300"+HEX$(teach_point_no)        /* query POSR
  GOSUB *RTU_QUERY_SND_RCV 8
  CMD$="0105040C0000"                           /* reset query CSTR
  GOSUB *RTU_QUERY_SND_RCV 8
  CMD$="0105040CFF00"                           /* set query CSTR
  GOSUB *RTU_QUERY_SND_RCV 8
  /* if CSTR=0xFF then doesn't activate PEND status
  CMD$="0105040C0000"                           /* reset query CSTR
  GOSUB *RTU_QUERY_SND_RCV 8
  GOSUB *WAIT_MOVE_END &H08                     /* &H8=PEND status
  RETURN

'=====================================
' Go to the origin
'=====================================
*HOME
  CMD$="0105040B0000"                           /* reset query HOME
  GOSUB *RTU_QUERY_SND_RCV 8
  CMD$="0105040BFF00"                           /* set query HOME
  GOSUB *RTU_QUERY_SND_RCV 8
  CMD$="0105040B0000"                           /* reset query HOME
  GOSUB *RTU_QUERY_SND_RCV 8
  GOSUB *WAIT_MOVE_END &H10                    /* &H10=HEND status
  RETURN

'=====================================
' reading the DSS1 and checking status
'=====================================
*WAIT_MOVE_END
_VAR chek_bit
/* chek_bit &H0010=HEND, &H0008=PEND
  DO
    GOSUB *CHECK_DSS1
    IF DSS1_STAT&chek_bit<>0 THEN
      BREAK
    END_IF
  LOOP
  RETURN

'=====================================
' reading the DSS1 and checking status
'=====================================
*WAIT_MOVE_END_A
_VAR chek_bit
/* chek_bit &H0010=HEND, &H0008=PEND
  DO
    GOSUB *CHECK_DSS1
    IF DSS1_STAT&chek_bit<>0 THEN
      BREAK
    ELSE
      GOSUB *DISPLAY_CURRENT_POSITION           /* display a current position
    END_IF
  LOOP
  RETURN

'=====================================
' Send query and response receive
'=====================================
*RTU_QUERY_SND_RCV
_VAR res_len                                    /* response length
  *RTU_QUERY_SND_RCV_RETRY
  DATA_CNT=LEN(CMD$)/2
  query_p=CMD$
  FOR query_i=1 TO DATA_CNT
    ptr_=query_p
    QUERY(query_i)=HEX(PTR$(2))                 /* Note: The HEX command changes the ptr_
    query_p=query_p+2
```

```
  NEXT
  GOSUB *RTU_CRC_CALC                          /* arguments: QUERY(), DATA_CNT
  QUERY(DATA_CNT+1)=CRC_L                      /* return value of *RTU_CRC_CALC
  QUERY(DATA_CNT+2)=CRC_H                      /* return value of *RTU_CRC_CALC
  SEND$=""                                     /* The each string size is 256bytes
  query_p=SEND$                                /* p is a pointer of SEND$
  FOR query_i=1 TO DATA_CNT+2
    POKE QUERY(query_i) query_p                /* create a binary data
    INC query_p
  NEXT
  PRINT# RS485_PORT STR_LEN|DATA_CNT+2 SEND$
  IF MONI_PORT<>0 THEN                         /* monitor output for debugging
    PRINT# MONI_PORT STR_LEN|DATA_CNT+2 SEND$
  END_IF

'=====================================
'Response processing
'=====================================
  RES$=""
  FORMAT "00"
  FOR query_i=1 TO res_len
    INPUT# RS485_PORT CHR_C|1 TMOUT|3 buf$
    IF rse_<>0 THEN
      ALERT_MSG$="res Time out"
      GOSUB *DISPLAY_ALERT
      GOTO *RTU_QUERY_SND_RCV_RETRY            /* retry
    END_IF
    IF MONI_PORT<>0 THEN                       /* monitor output for debugging
      PRINT# MONI_PORT STR_LEN|1 buf$
    END_IF
    QUERY(query_i)=ASC(buf$)
    RES$=RES$+HEX$(QUERY(query_i))
    'PRINT "RES(" array_num ")=" HEX$(RES(array_num))
  NEXT
  DATA_CNT=res_len-2
  GOSUB *RTU_CRC_CALC                          /* arguments: QUERY(), DATA_CNT
  'PRINT "CRC L H" HEX$(CRC_L) HEX$(CRC_H)
  IF (QUERY(DATA_CNT+1)<>CRC_L) OR (QUERY(DATA_CNT+2)<>CRC_H) THEN /* CRC compare RES to CACL
    PRINT "CRC L ERROR" HEX$(QUERY(DATA_CNT+1)) HEX$(CRC_L)
    FORMAT "00"
    ALERT_MSG$="CRC err "+HEX$(QUERY(DATA_CNT+1))+","+HEX$(CRC_L)
    ALERT_MSG$=ALERT_MSG$+"/"+HEX$(QUERY(DATA_CNT+2))+","+HEX$(CRC_H)
    GOSUB *DISPLAY_ALERT
    GOTO *RTU_QUERY_SND_RCV_RETRY              /* retry
  END_IF
  TIME 5
  RETURN

'=====================================
' CRC16 calculate
'=====================================
*RTU_CRC_CALC
  CRC=&HFFFF
  FOR crc_i=1 TO DATA_CNT
    crc_next=QUERY(crc_i)
    CRC=(CRC^crc_next)&&HFFFF
    FOR crc_j=1 TO 8
      crc_carry=CRC&&H1
      CRC=CRC/2
      IF crc_carry=1 THEN
        CRC=(CRC^&HA001)&&HFFFF
      END_IF
      SWAP
    NEXT crc_j
    CRC_L=CRC&&HFF
    CRC_H=(CRC&&HFF00)/256
  NEXT crc_i
  RETURN

'=====================================
' An error message pop-up on the touch panel display
'=====================================
*DISPLAY_ALERT
  previous_screen=MBK(0)                       /* previous screen no.
  S_MBK ALERT_MSG$ 360 20                      /* display alert message
  OFF 71500
  S_MBK 7 0                                    /* open the alert window
  WAIT SW(71500)==1                            /* notification of the window close
  OFF 71500
  S_MBK previous_screen 0
  RETURN
```

--End Of File--