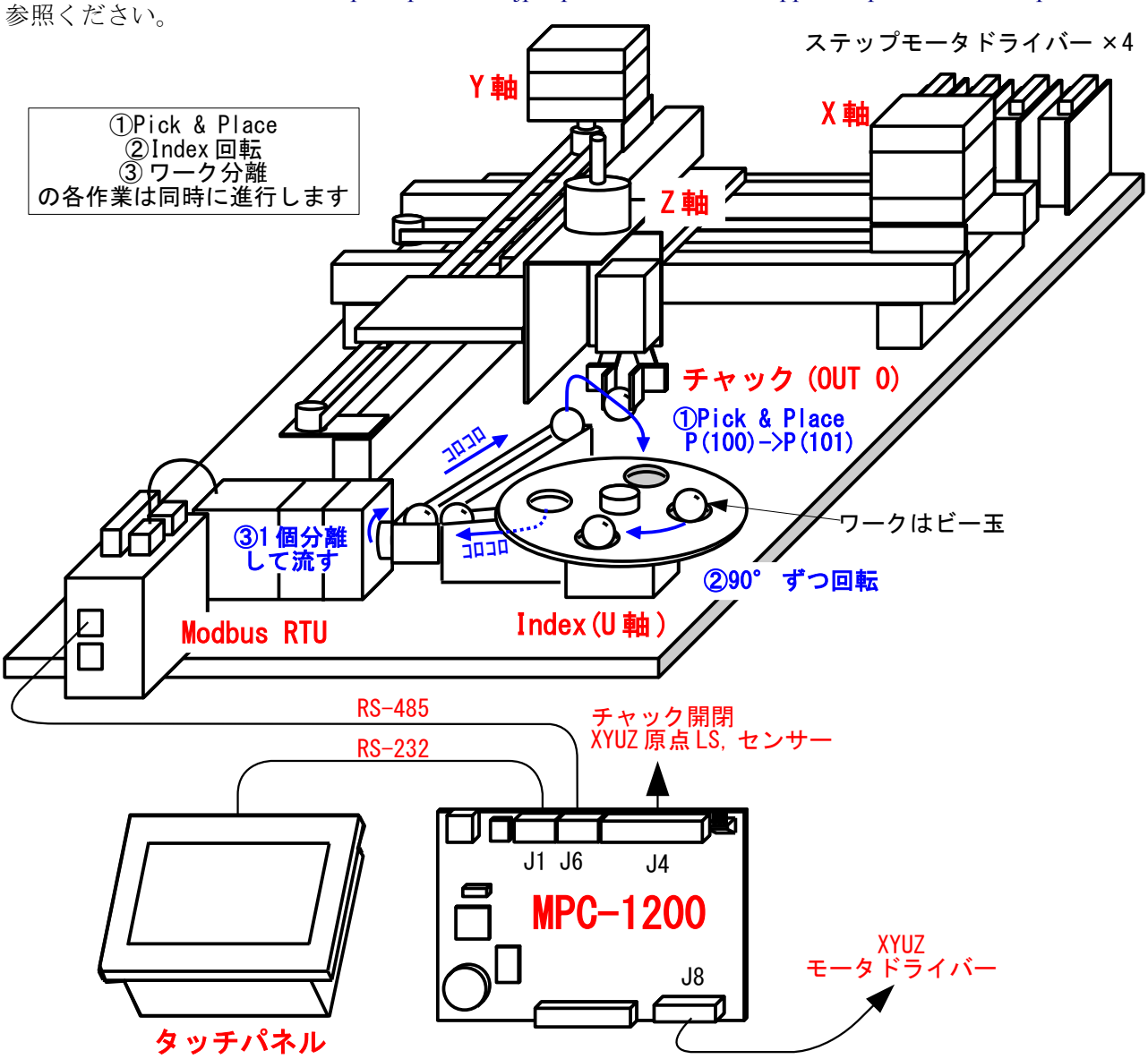
Application Note	Ref No: an2k-051	Last Modify 161201
テーマ	諏訪圏工業メッセ 2016 展示品(MPC-1200 単体 5 軸制御)	
使用機器	MPC-1200	

構成

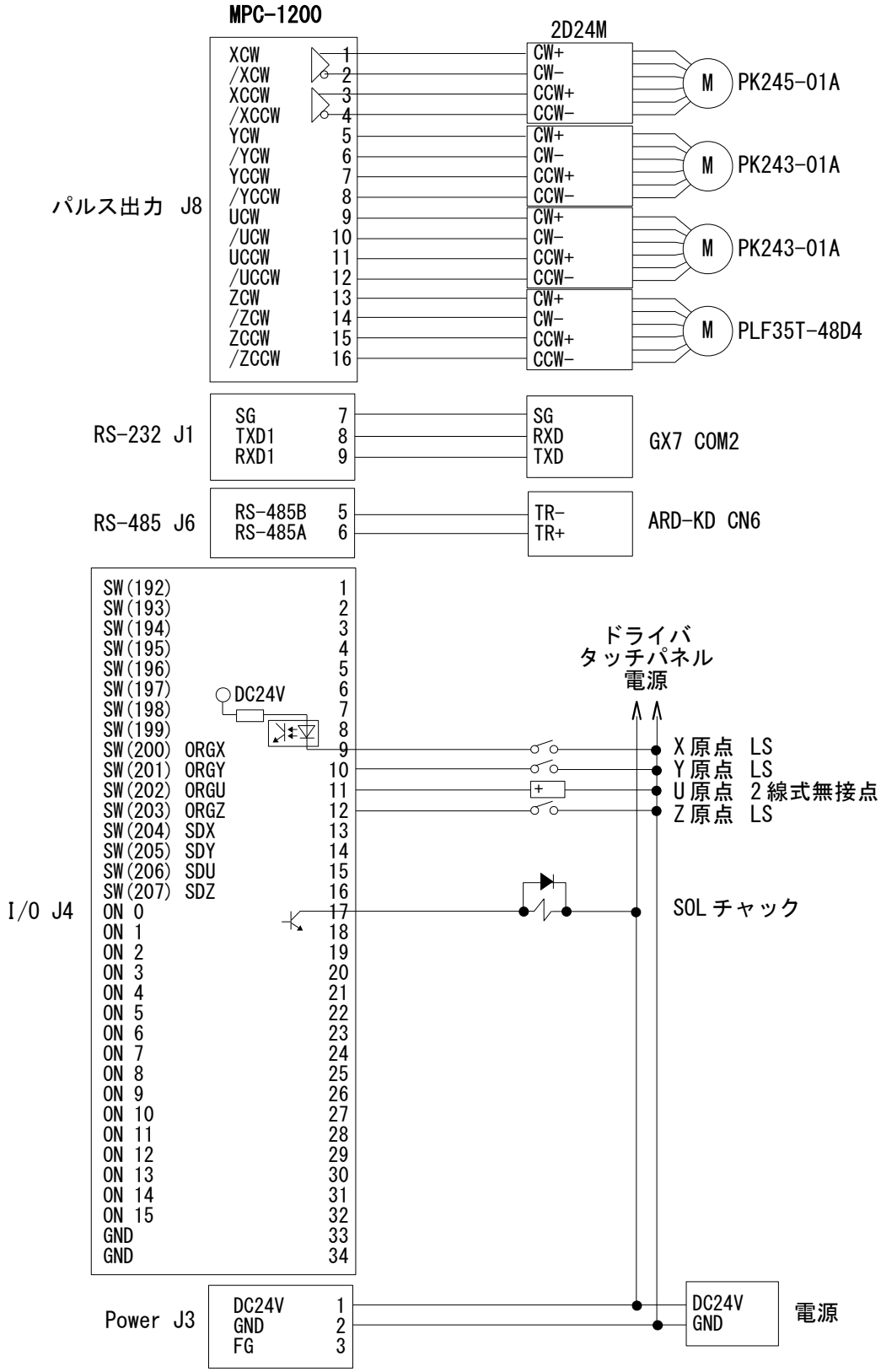
諏訪圏工業メッセ 2016 に展示したデモ機です。MPC-1200 1 台で XYZ ロボット＋インデックステーブル＋Modbus RTU の 5 軸を制御してビー玉を循環させます。ワーク搬送をイメージしました。

Modobus RTU コマンドは http://departonline.jp/mpc2000/ref/headline/appendix/pdf/ti2k-160113.pdf を御参照ください。



X 軸	Orientalmotor	PK245-01A	1/60 マイクロステップ駆動
Y 軸	Orientalmotor	PK243-01A	1/60 マイクロステップ駆動
Z 軸	NPM	PLF35T-48D4	1/25 マイクロステップ駆動
U 軸	Orientalmotor	Orientalmotor	1/25 マイクロステップ駆動
Modbus-RTU	Orientalmotor	ドライバ ARD-KD	2 点間の回転。ポジションは MPC のプログラム中で指定
		モータ ARM-46AK	
XYUZ 軸ドライバ	PPS	2D24M	
タッチパネル	MISUMI	PLCP-GX7-05-DC-R	

結線


MPC-1200
2D24M
パルス出力 J8
XCW 1
/XCW 2
XCCW 3
/XCCW 4
YCW 5
/YCW 6
YCCW 7
/YCCW 8
UCW 9
/UCW 10
UCCW 11
/UCCW 12
ZCW 13
/ZCW 14
ZCCW 15
/ZCCW 16
CW+
CW−
CCW+
CCW−
M
PK245-01A
PK243-01A
PK243-01A
PLF35T-48D4
RS-232 J1
SG 7
TXD1 8
RXD1 9
SG
RXD
TXD
GX7 COM2
RS-485 J6
RS-485B 5
RS-485A 6
TR−
TR+
ARD-KD CN6
I/O J4
SW(192) 1
SW(193) 2
SW(194) 3
SW(195) 4
SW(196) 5
SW(197) 6
SW(198) 7
SW(199) 8
SW(200) ORGX 9
SW(201) ORGY 10
SW(202) ORGU 11
SW(203) ORGZ 12
SW(204) SDX 13
SW(205) SDY 14
SW(206) SDU 15
SW(207) SDZ 16
ON 0 17
ON 1 18
ON 2 19
ON 3 20
ON 4 21
ON 5 22
ON 6 23
ON 7 24
ON 8 25
ON 9 26
ON 10 27
ON 11 28
ON 12 29
ON 13 30
ON 14 31
ON 15 32
GND 33
GND 34
DC24V
ドライバ
タッチパネル
電源
X原点 LS
Y原点 LS
U原点 2線式無接点
Z原点 LS
SOL チャック
Power J3
DC24V 1
GND 2
FG 3
DC24V
GND
電源

MPC Program

```
  QUIT_FORK 1 *main
  END
*main
  MEWNET 38400 1
  MODBUS 2 2 "38400b8pns1NONE"                    /* Modbus Port 初期化

  QUIT_FORK 2 *FLIP
  QUIT_FORK 3 *Coordinates

  MBK(2)=200                                      /* GX7 Page Control
  MBK(208)=0                                      /* Job Count

  OFF 70000                                       /* Start/Pause オルターネート SW
  OFF 0
  OFF -1
  PG 17

  WAIT SW(70000)==1                               /* タッチパネル Start SW

  DO

    GOSUB *HOME
    U(100)=0
    U(101)=0

    ACCEL X_A|Y_A 120000 12000 500                /* XY 軸加減速、初速
    ACCEL Z_A 60000 3000 100                      /* Z 軸加減速、初速
    ACCEL U_A 7000 600 200                        /* U 軸加減速、初速
    FEED ALL_A 100

    cnt=0
    DO
      WAIT SW(70000)==1
      CLRPOS U_A

      MOVS Z_A Z(100)                             /* XY が Z より十分に速い場合
      MOVL X(100) Y(100) 1250
      WAIT RR(ALL_A)==0

      MOVL X(100) Y(100) VOID Z(100)              /* 給材点
      WAIT RR(ALL_A)==0

      WAIT SW(70000)==1

      ON 0                                        /* Close Chuck
      TIME 200

      MOVS Z_A -10000                             /* ワークを投下する高さ
      WAIT Z(0)>(Z(100)+10000)
      ON -1                                       /* Tilt Flip
      WAIT Z(0)>-25000                            /* Z 移動途中で XY 移動開始
      MOVL X(101) Y(101)                          /* Index 除材点
      WAIT RR(ALL_A)==0

      WAIT SW(70000)==1

      OFF 0
      TIME 100

      RMVS Y_A 5000                               /* Y 軸横に退避
      WAIT RR(ALL_A)==0

      MBK(208)=MBK(208)+1
      cnt=cnt+1
```

```
      IF cnt>12 THEN
        BREAK
      END_IF

    LOOP

    MOVS Z_A -1000
    WAIT RR(Z_A)==0
    MOVL 20000 20000
    WAIT RR(ALL_A)==0
    WAIT SW(70000)==1

  LOOP

*HOME                                   /* XYZU 軸原点復帰
  SHOM &H55
  ACCEL X_A|Y_A 10000 1000 500
  ACCEL Z_A 10000 500 100
  ACCEL U_A 1500 500 250

  IF SW(203)==1 THEN
    RMVS Z_A -10000
    WAIT RR(Z_A)==0
  END_IF

  HOME Z_A POS_L                        /* Z 軸原点復帰
  WAIT RR(ALL_A)==0

  IF SW(202)==1 THEN                    /* U 軸 Org
    RMVS U_A -1000                      /* U 軸退避移動
  END_IF
  IF SW(200)==1 THEN
    RMVS X_A 5000                       /* X 軸退避移動
  END_IF
  IF SW(201)==1 THEN
    RMVS Y_A 5000                       /* Y 軸退避移動
  END_IF
  WAIT RR(ALL_A)==0
  TIME 500

  HOME X_A|Y_A NEG_L
  HOME U_A POS_L                        /* U 軸原点復帰
  WAIT RR(ALL_A)==0

  RMVS X_A|Y_A 1000                     /* XY 軸 Offset
  RMVS Z_A -5000                        /* Z 軸 Offset
  RMVS U_A 690                          /* U 軸 Offset
  WAIT RR(ALL_A)==0
  CLRPOS
  TIME 100

  RETURN

*FLIP                                   /* Modbus RTU
  GOSUB *ARD_SET_PARAM                  /* ARD パラメータ設定
  GOSUB *ARD_WRITE_REG_L &H7D &H0       /* ARD コマンドレジスタ リセット
  GOSUB *ARD_POINT_SET                  /* ARD ポイント設定

  GOSUB *ARD_HOME1                      /* ARD 原点復帰
  GOSUB *ARD_POINT_MOVE 0               /* ARD ポイント移動

  DO
    WAIT SW(-1)==1
    IF SW(70001)==0 THEN                /* IF SW(70001)==1 THEN Tilt Cancel
      GOSUB *ARD_POINT_MOVE 1           /* ARD ポイント移動
      TIME 10
```

```
      GOSUB *ARD_POINT_MOVE 0                    /* ARD ポイント移動
    ELSE
      OFF 70001
    END_IF
    OFF -1
  LOOP

*FLIP_TO_P11                                     /* 調整用ルーチン
  GOSUB *ARD_POINT_MOVE 1                        /* ARD ポイント移動
  END

*Coordinates                                     /* タッチパネル座標表示
  PG 17
  DO
    MBK(200~Lng)=X(0)
    MBK(202~Lng)=Y(0)
    MBK(204~Lng)=Z(0)
    MBK(206~Lng)=U(0)
    SWAP
  LOOP


'****************************************
' オリエンタルモーター ARD-KD
' 駆動サブルーチン
'****************************************

'======================================
' ARD 点移動
'======================================
*ARD_POINT_MOVE
_VAR Ard_PointNum

  prc$="ARD_POINT_MOVE" : PR prc$ Ard_PointNum

  GOSUB *ARD_WRITE_REG_W &H007D &H8|Ard_PointNum   /* ドライバ入力指令 Start|(M2,M1,M0)
  GOSUB *ARD_WRITE_REG_W &H007D &H0
  GOSUB *ARD_WAIT_READY
  GOSUB *ARD_CURRENT_POS

  RETURN

'======================================
' ARD Jog 移動
'======================================
*ARD_JOG
_VAR Adr_JogDistance

  prc$="ARD_JOG" : PR prc$ Adr_JogDistance

  GOSUB *ARD_WRITE_REG_L &H0288 1000             /* JOG 加減速
  GOSUB *ARD_WRITE_REG_L &H028A 100              /* JOG 起動速度
  GOSUB *ARD_WRITE_REG_L &H0286 2000             /* JOG 運転速度Hz
  GOSUB *ARD_WRITE_REG_L &H1048 Adr_JogDistance /* JOG 移動量
  GOSUB *ARD_WRITE_REG_W &H007D &H2000           /* ドライバ入力指令 -JOG
  GOSUB *ARD_WRITE_REG_W &H007D &H0
  GOSUB *ARD_WAIT_READY

  RETURN

'======================================
' ARD 点の設定
'======================================
*ARD_POINT_SET

  prc$="ARD_POINT_SET" : PR prc$
```

```
/* P0
    GOSUB *ARD_WRITE_REG_L &H0400+0 10          /* 位置
    GOSUB *ARD_WRITE_REG_L &H0480+0 2000        /* 運転速度
    GOSUB *ARD_WRITE_REG_L &H0500+0 1           /* 0:インクリメンタル, 1:アブソリュート
    GOSUB *ARD_WRITE_REG_L &H0580+0 0           /* 0:単軸, 1:連結
    GOSUB *ARD_WRITE_REG_L &H0600+0 1000        /* 加速
    GOSUB *ARD_WRITE_REG_L &H0680+0 1000        /* 減速

/* P1
    GOSUB *ARD_WRITE_REG_L &H0400+2 100         /* 位置
    GOSUB *ARD_WRITE_REG_L &H0480+2 2000        /* 運転速度
    GOSUB *ARD_WRITE_REG_L &H0500+2 1           /* 0:インクリメンタル, 1:アブソリュート
    GOSUB *ARD_WRITE_REG_L &H0580+2 0           /* 0:単軸, 1:連結
    GOSUB *ARD_WRITE_REG_L &H0600+2 1000        /* 加速
    GOSUB *ARD_WRITE_REG_L &H0680+2 1000        /* 減速

  RETURN

'====================================
' ARD 原点復帰
'====================================
*ARD_HOME

  prc$="ARD_HOME" : PR prc$

  GOSUB *ARD_WRITE_REG_L &H02C2 500             /* 原点復帰速度
  GOSUB *ARD_WRITE_REG_L &H02C6 100             /* 原点復帰起動速度
  GOSUB *ARD_WRITE_REG_L &H02C4 200             /* 原点復帰加減速
  GOSUB *ARD_WRITE_REG_L &H02CA 0               /* 原点復帰開始方向 0:-側
  GOSUB *ARD_WRITE_REG_W &H007D &H10            /* ドライバ入力指令 Home
  GOSUB *ARD_WRITE_REG_W &H007D &H0
  GOSUB *ARD_WAIT_READY
  GOSUB *ARD_CURRENT_POS
  RETURN

'====================================
' ARD 原点復帰
'====================================
*ARD_HOME1

  prc$="ARD_HOME" : PR prc$

  GOSUB *ARD_WRITE_REG_L &H02C2 500             /* 原点復帰速度
  GOSUB *ARD_WRITE_REG_L &H02C6 100             /* 原点復帰起動速度
  GOSUB *ARD_WRITE_REG_L &H02C4 200             /* 原点復帰加減速
  GOSUB *ARD_WRITE_REG_L &H02CA 1               /* 原点復帰開始方向 1:+側
  GOSUB *ARD_WRITE_REG_W &H007D &H10            /* ドライバ入力指令 Home
  GOSUB *ARD_WRITE_REG_W &H007D &H0
  GOSUB *ARD_WAIT_READY
  GOSUB *ARD_CURRENT_POS
  RETURN


'====================================
' ARD レディー待ち
'====================================
*ARD_WAIT_READY

  prc$="ARD_WAIT_READY" : PR prc$

  DO
    GOSUB *ARD_READ_REG_W &H7F                  /* ドライバ出力指令
    IF R_RTU(Byt,5)&&H20<>0 THEN                /* 受信データの5バイト目からByt(1バイト)
読込
```

```
      BREAK
    END_IF
  LOOP
  RETURN

'=====================================
' ARD 現在位置取得
'=====================================
*ARD_CURRENT_POS

  prc$="ARD_CURRENT_POS" : PR prc$

  GOSUB   *ARD_READ_REG_L &HCC
    Ard_CurPos=R_RTU(Lng,4)                    /* 受信データの4バイト目から4バイト読込
    PR "ARD Current Position=" Ard_CurPos
  RETURN

'=====================================
' ARD パラメータセット
'=====================================
*ARD_SET_PARAM

  prc$="ARD_SET_PARAM" : PR prc$

  GOSUB *ARD_WRITE_REG_L &H21A 0               /* HOMES 接点設定 0:N.O.
  GOSUB *ARD_WAIT_READY
  GOSUB *ARD_WRITE_REG_L &H18C 1               /* Configuration
  GOSUB *ARD_WRITE_REG_L &H18C 0
  GOSUB *ARD_WAIT_READY
  RETURN

'=====================================
' ARD 2バイト書き込み
'=====================================
*ARD_WRITE_REG_W
_VAR  Ard_RegAdd Ard_WriteData

/* 保持レジスタへの書き込み(2バイト長)
/* Query Example
/* 0206024B0050xxxx
/* 02 スレーブアドレス
/*   06 ファンクションコード
/*     024B 書き込みを行うレジスタアドレス (Wrd)
/*         0050 レジスタに書き込む値 (Wrd)
/*             xxxx CRC
/*
/* Responce Example
/* 0206024B0050xxxx
/* 02 スレーブアドレス              ┐
/*   06 ファンクションコード        │ 6バイト
/*     024B レジスタアドレス (Wrd)  │
/*         0050 ライト値 (Wrd)      ┘
/*             xxxx CRC

  DO
    TIME 5 : TMOUT 2000                        /* Silent Interval : Time Out
    X_RTU 2 &H06 Ard_RegAdd~Wrd Ard_WriteData~Wrd
    r_res=R_RTU(6)
    IF r_res==1 THEN
      BREAK
    ELSE
     GOSUB *ERROR_DISP
    END_IF
  LOOP
  RETURN
```

```
'====================================
' ARD 4バイト書き込み
'====================================
*ARD_WRITE_REG_L
_VAR  Ard_RegAdd Ard_WriteData

/* 複数の保持レジスタへの書き込み(4バイト長)
/* Query Example
/* 021002C2000204000001F4xxxx
/* 02 スレーブアドレス
/*   10 ファンクションコード
/*     02C2 書き込みの起点となるレジスタアドレス (Wrd)
/*         0002 起点のレジスタアドレスから書き込むレジスタの数 (Wrd)
/*             04 バイト数 (Byt)
/*               000001F4 レジスタアドレス 02C2h, 02C3h の書き込み値 (Lng)
/*                       xxxx  CRC
/*
/* Responce Example
/* 021002C20002xxxx
/* 02 スレーブアドレス                ┐
/*   10 ファンクションコード          │ 6バイト
/*     02C2 レジスタアドレス (Wrd)    │
/*         0002 レジスタの数 (Wrd)    ┘
/*             xxxx CRC

  DO
    TIME 5 : TMOUT 2000                  /* Silent Interval : Time Out
    X_RTU 2 &H10 Ard_RegAdd~Wrd 2~Wrd 4~Byt Ard_WriteData   /* クエリー送信
    r_res=R_RTU(6)
    IF r_res==1 THEN
      BREAK
    ELSE
     GOSUB *ERROR_DISP
    END_IF
  LOOP
  RETURN


'====================================
' ARD 2バイト読み出し
'====================================
*ARD_READ_REG_W
_VAR  Ard_RegAdd

/* 保持レジスタの読み出し(2バイト長)
/* Query Example
/* 0203007F0001xxxx
/* 02 スレーブアドレス
/*   03 ファンクションコード
/*     007F 読み出しの起点となるレジスタアドレス (Wrd)
/*         0001 起点のレジスタアドレスから読み出すレジスタの数 (Wrd)
/*             xxxx  CRC
/*
/* Responce Example
/* 0203021234xxxx
/* 02 スレーブアドレス                                    ┐
/*   03 ファンクションコード                              │ 5バイト
/*     02 データバイト数 (Byt)                            │
/*       1234 レジスタアドレス 007F の読み出し値 (Wrd)    ┘
/*           xxxx CRC

  DO
    TIME 5 : TMOUT 2000                        /* Silent Interval : Time Out
    X_RTU 2 &H03 Ard_RegAdd~Wrd 1~Wrd
    r_res=R_RTU(5)
    IF r_res==1 THEN
```

```
      BREAK
    ELSE
     GOSUB *ERROR_DISP
    END_IF
  LOOP
  RETURN

'====================================
' ARD 4バイト読み出し
'====================================
*ARD_READ_REG_L
_VAR   Ard_RegAdd

/* 保持レジスタの読み出し(4バイト長)
/* Query Example
/* 020300CC0002xxxx
/* 02 スレーブアドレス
/*   03 ファンクションコード
/*     00CC 読み出しの起点となるレジスタアドレス (Wrd)
/*         0002 起点のレジスタアドレスから読み出すレジスタの数 (Wrd)
/*             xxxx  CRC
/*
/* Responce Example
/* 02030412345678xxxx
/* 02 スレーブアドレス                                                    ┐
/*   03 ファンクションコード                                              ├ 7バイト
/*     04 データバイト数 (Byt)                                            ┘
/*       12345678 レジスタアドレス 00CC, 00CD の読み出し値 (Lng)
/*               xxxx CRC

  DO
    TIME 5 : TMOUT 2000                 /* Silent Interval : Time Out
    X_RTU  2 &H03 Ard_RegAdd~Wrd 2~Wrd
    r_res=R_RTU(7)                      /* 受信バイト数を指定する(CRCは含まない)
    IF r_res==1 THEN
      BREAK
    ELSE
     GOSUB *ERROR_DISP
    END_IF
  LOOP
  RETURN


'====================================
' エラー表示
'====================================
*ERROR_DISP
  oldpage=MBK(2)
  MBK(2)=7                                      /* タッチパネルエラー表示画面
  FORMAT ""
  SELECT_CASE r_res                             /* r_resはR_RTUの戻り値
    CASE -1 : errmsg$=prc$+" TIME OUT"
    CASE 0  : errmsg$=prc$+" CRC ERROR"
    CASE_ELSE : errmsg$=prc$+" ?"
  END_SELECT
  PR errmsg$

  S_MBK errmsg$ 360 30
  WAIT SW(71500)==1                             /* Restartボタン
  MBK(2)=oldpage
  TMOUT 2000                                    /* 通信タイムアウト再設定
  RETURN
```