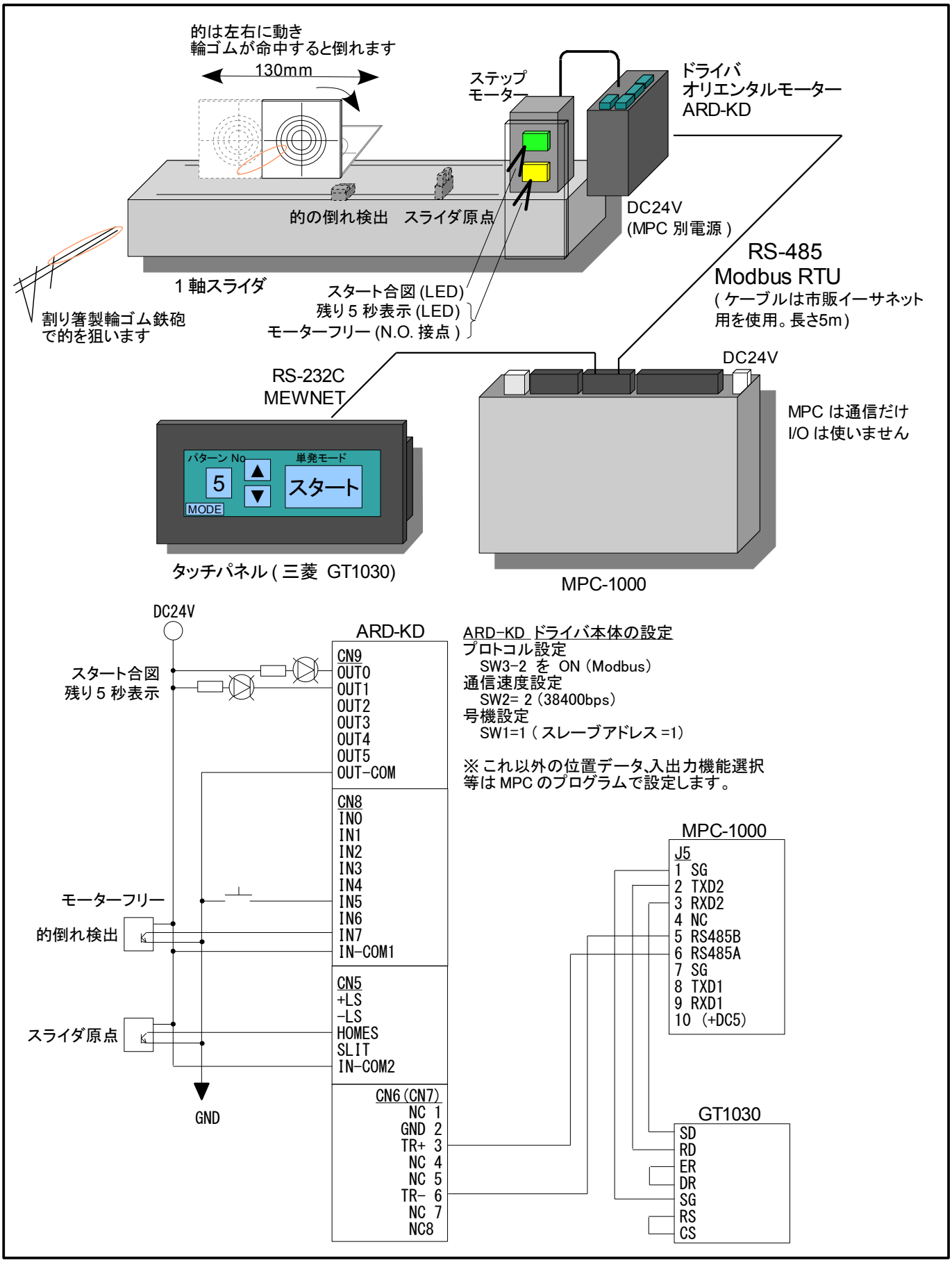
Application Note		資料作成 101028	資料番号 an2k-024
テーマ	MPC-1000 Modbus 通信制御 (諏訪圏工業メッセ 2010 出展 輪ゴム鉄砲的当てゲーム)		
使用機器	MPC-1000、オリエンタルモータ ARD-KD(ドライバ・モータセット)、タッチパネル		

■機器構成



オリエンタルモータ㈱製 ARD-KD はコントローラ内臓・RS-485 機能付のステップモータドライバです。
Modbus： Modicon 社によって開発された オープン Master/Slave アプリケーションプロトコル
RTU：Remote Terminal Unit (遠隔端末装置)
CRC-16:Cyclic Redundancy Check 16 (巡回冗長検査)

■MPC プログラム

【動作概要】

- ARD には位置 No0～7 の座標とそれぞれの移動速度を設定します。それら各点間移動と JOG 移動を組み合わせて 0~9 の 10 種の動作パターンを作成します。0 から順に難易度が高くなります。
- 動作モードには 単発モードと連続モードがあり、タッチパネルで切り替えます。単発モードはタッチパネルの DSW でパターン番号を選びスタートボタン押下で一回ずつ実行します。連続モードはストップスイッチを押すまでパターンを昇順に連続実行します。
- 1 パターン約 20 秒で 5 秒前になると黄色 LED が点灯します。時間内に輪ゴムが的に当たって倒れればタッチパネルに経過時間を表示します。
- 通信チェックは CRC-16 です。
- モーターフリースイッチは MPC で制御していません。スイッチを押している間モーターはフリー状態となりスライダーを手で動かすことができます。ARD にはアブソリュートエンコーダが内臓されておりフリー時でも位置を認識しています。

- 以下出展時のプログラムそのまま(デバッグ用の通信やサブルーチン等も含まれます)

```
*_BEGIN
/* ARD COMMAND register / MPC memory I/O table
/* ------------------------------------------------------------
/* &H7D(high)|NET-IN15|-IN14|-IN13|-IN12|-IN11|-IN10|-IN9 |-IN8 |
/* signal    |        |      |      |      |      |      |      | R0  |
/* mem I/O   |  -16   | -15  | -14  | -13  | -12  | -11  | -10  |  -9 |
/* ------------------------------------------------------------
/* &H7D(low) |NET-IN7 |-IN6  |-IN5  |-IN4  |-IN3  |-IN2  |-IN1  |-IN0 |
/* signal    |  -JOG  |+JOG  |STOP  |HOME  |START |  M2  |  M1  |  M0 |
/* mem I/O   |  -8    |  -7  |  -6  |  -5  |  -4  |  -3  |  -2  |  -1 |
/* ------------------------------------------------------------

  DIM QUERY(16)                 /* using QUERY(1)~
  DIM RES(16)                   /* using RES(1)~
  DIM BUFF(16)                  /* using BUFF(1)~

  FILL X(1000) 1000 -1

  MBK(1000)=1000                /* P0,1   SPEED
  MBK(1001)=2000                /* P0,1   accel/decel
  MBK(1002)=5000                /* P2,3,4   SPEED
  MBK(1003)=2000                /* P2,3,4   accel/decel
  MBK(1004)=40000               /* P5,6,7   SPEED
  MBK(1005)=2000                /* P5,6,7   accel/decel

  MBK(1010)=4000                /* JOG speed Hz
  MBK(1011)=2000                /* JOG acceleration/deceleration
  MBK(1012)=50                  /* JOG start speed

*PAT0
  SETP 1000 1 0 0 0
  SETP 1001 0 0 0 0

*PAT1
  SETP 1100 4 0 0 0
  SETP 1101 2 0 0 0

*PAT2
  SETP 1200 3 0 0 0
  SETP 1201 4 0 0 0
  SETP 1202 3 0 0 0
  SETP 1203 2 0 0 0

*PAT3
  SETP 1300 7 0 0 0
  SETP 1301 5 0 0 0

*PAT4
  SETP 1400 6 0 0 0
  SETP 1401 7 0 0 0
  SETP 1402 6 0 0 0
```

```
  SETP 1403 5 0 0 0

*PAT5
  SETP 1500 8 0 0 0
  SETP 1501 8 0 0 0
  SETP 1502 2 0 0 0

*PAT6
  SETP 1600 8 0 0 0
  SETP 1601 8 0 0 0
  SETP 1602 9 0 0 0
  SETP 1603 9 0 0 0

*PAT7
  SETP 1700 7 0 0 0
  SETP 1701 9 0 0 0
  SETP 1702 5 0 0 0
  SETP 1703 8 0 0 0

*PAT8
  SETP 1800 7 0 0 0
  SETP 1801 6 0 0 0
  SETP 1802 4 0 0 0
  SETP 1803 5 0 0 0
  SETP 1804 6 0 0 0
  SETP 1805 2 0 0 0

*PAT9
  SETP 1900 7 0 0 0
  SETP 1901 6 0 0 0
  SETP 1902 7 0 0 0
  SETP 1903 5 0 0 0
  SETP 1904 6 0 0 0
  SETP 1905 5 0 0 0

  CONST   PAGING 8
  ARD_STOP=0

  MEWNET 38400 2 B80
  CNFG# RS485 1 "38400b8pes1NONE"               /* RS485 initialize
  CNFG# 5 "38400b8pes1NONE"                     /* monitor for debug
  S_MBK 200 PAGING


  TIME 1000
  slave_address=1                               /* slave address

  IF SW(192)==0 THEN
    GOSUB *SET_NET-IN-OUT_PARAM
    'END
  END_IF

  OUT 0 -1~Wrd
  GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)        /* command register reset
  /*GOSUB *QUERY_WRITE_REG &H7D &H0             /* command register reset
  GOSUB *SET_POINT_PARAM
  GOSUB *SET_SPEED_PARAM

  QUIT 1
  FORK 1 *MOVE
  END

*MOVE

  DO

    GOSUB *R0_OFF
    GOSUB *R1_OFF
    GOSUB *HOME
    OFF 70002

    IF MBK(PAGING)<>204 THEN                    /* not 連続 page
      S_MBK 201 PAGING                          /* 単発 page
      OFF 70000                                 /* start button
    END_IF
```

```
*move_1
DO                                          /* wait for start

  /*IF MBK(PAGING)==201 THEN                /* if demo mode then doesn't wait for start
  WAIT SW(70000)==1
  /*END_IF

  GOSUB *QUERY_READ_REG &H7F
  IF RES(5)&&H1==0 THEN                     /* if target is falling down
    GOSUB *HOME
  END_IF

  DSW=MBK(100)
  PAT_TOP=DSW*100+1000
  IF X(PAT_TOP)<>-1 THEN                    /* data check
    BREAK                                   /* exit from DO~LOOP
  END_IF
  PRINT "Pattern No Error"
  OFF 70000
LOOP

OUT &H8 -1                                  /* [start | M2 M1 M0] -> [out -1] ; move to point0
GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)
OUT 0 -1
GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)
GOSUB *WAIT_FOR_READY_STOP

GOSUB *R1_OFF
GOSUB *R0_ON

FOR J=0 TO 100                              /* searching last of data
  IF X(PAT_TOP+J)==-1 THEN
    BREAK
  END_IF
NEXT
J=J-1                                       /* last No of the Pattern

*move_2
ARD_STOP=0
R1_ON=0
SYSCLK=0
F=1
WHILE F==1

  FOR I=0 TO J

    DEST_POINT=X(PAT_TOP+I)
    SELECT_CASE DEST_POINT
      CASE 8 : GOSUB *JOG+
      CASE 9 : GOSUB *JOG-
      CASE_ELSE
        OUT &H8|DEST_POINT -1               /* [start | M2 M1 M0] -> [out -1]
        GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)
        OUT 0 -1
        GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)
        GOSUB *WAIT_FOR_READY_STOP
        'GOSUB *READ_CURRENT_POS
    END_SELECT

    IF ARD_STOP<>0 THEN
      F=0
      BREAK                                 /* exit from for~next
    END_IF

    IF (SYSCLK>15000)&(R1_ON==0) THEN
      GOSUB *R1_ON
      GOSUB *R0_OFF
      R1_ON=1
    END_IF
    TIME Y(PAT_TOP+I)

  NEXT I

  IF SYSCLK>20000 THEN
```

```
    ARD_STOP=0
    F=0
  END_IF

  IF SW(70002)==1 THEN                      /* exit from demo mode
    ARD_STOP=0
    F=0                                     /* exit from WHILE
  END_IF

WEND

*move_3
/*IF (MBK(PAGING)<>204)&(SW(70002)==0) THEN   /* not 連続 page
IF MBK(PAGING)==201 THEN                    /* 単発 page
  IF ARD_STOP<>0 THEN                       /* process of results
    timer_=50
    DO
      S_MBK ARD_STOP/10 101                 /* elapsed time
      S_MBK 202 PAGING                      /* 当たり page
      GOSUB *R0_ON                          /* LED flick
      GOSUB *R1_ON
      TIME 50
      GOSUB *R0_OFF
      GOSUB *R1_OFF
      TIME 50

      IF timer_<=0 THEN
        S_MBK 200 PAGING                    /* logo page
        BREAK
      END_IF
    LOOP
  ELSE
    PRINT SYSCLK
    S_MBK 203 8                             /* 残念 page
    GOSUB *R0_OFF
    GOSUB *R1_OFF
    TIME 5000
    S_MBK 200 PAGING                        /* logo page
  END_IF
END_IF

*move_4
IF MBK(PAGING)==204 THEN                    /* 連続 page

  IF (ARD_STOP<>0)&(SW(70002)==0) THEN      /* process of results
    timer_=50
    DO
      S_MBK ARD_STOP/10 101                 /* elapsed time
      S_MBK 202 PAGING                      /* 当たり page
      GOSUB *R0_ON                          /* LED flick
      GOSUB *R1_ON
      TIME 50
      GOSUB *R0_OFF
      GOSUB *R1_OFF
      TIME 50

      IF timer_<=0 THEN
        S_MBK 204 PAGING                    /* 連続 page
        BREAK
      END_IF
    LOOP
  ELSE
    GOSUB *R0_OFF
    GOSUB *R1_OFF
    IF SW(70002)==0 THEN
      S_MBK 203 PAGING                      /* 残念 page
      TIME 3000
    END_IF
    S_MBK 204 PAGING
  END_IF

  IF SW(70002)==1 THEN
    OFF 70000
    OFF 70002
  ELSE
```

```
        MBK(100)=MBK(100)+1
        IF MBK(100)>9 THEN
          MBK(100)=0
        END_IF
      END_IF

    END_IF

  LOOP

*JOG+
  FOR JOG_CNT=0 TO 3
    ON -7                                   /* +JOG
    GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)
    OFF -7                                  /* +JOG
    GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)
    GOSUB *WAIT_FOR_READY_STOP
    IF ARD_STOP<>0 THEN
      BREAK
    END_IF
  NEXT
RETURN

*JOG-
  FOR JOG_CNT=3 TO 0 STEP -1
    ON -8                                   /* -JOG
    GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)
    OFF -8
    GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)
    GOSUB *WAIT_FOR_READY_STOP
    IF ARD_STOP<>0 THEN
      BREAK
    END_IF
  NEXT
RETURN

*SET_NET-IN-OUT_PARAM

  GOSUB *QUERY_WRITE_REG_L &H116C 6         /* NET-IN6 = +JOG
  GOSUB *QUERY_WRITE_REG_L &H116E 7         /* NET-IN7 = -JOG
  GOSUB *QUERY_WRITE_REG_L &H1178 0         /* NET-IN12
  GOSUB *QUERY_WRITE_REG_L &H117A 0         /* NET-IN13

  GOSUB *QUERY_WRITE_REG_L &H1140 32        /* OUT0  = R0
  GOSUB *QUERY_WRITE_REG_L &H1170 32        /* NET-IN8 = R0
  GOSUB *QUERY_WRITE_REG_L &H1142 33        /* OUT1  = R1
  GOSUB *QUERY_WRITE_REG_L &H1172 33        /* NET-IN9 = R1

  GOSUB *QUERY_WRITE_REG_L &H110E 36        /* IN7  = R4
  GOSUB *QUERY_WRITE_REG_L &H1180 36        /* NET-OUT0 = R4

  GOSUB *QUERY_WRITE_REG_L &H21A 1          /* HOMES N.C.

  GOSUB *WAIT_FOR_READY                     /* need
  PRINT "ARD configuration start"
  GOSUB *QUERY_WRITE_REG_L &H18C 1          /* configuration
  GOSUB *QUERY_WRITE_REG_L &H18C 0          /*
  GOSUB *WAIT_FOR_READY
  PRINT "ARD configuration complete"

  RETURN

*SET_POINT_PARAM

  GOSUB *QUERY_WRITE_REG_L &H284 0          /* Start Speed

  FOR I=0 TO 7
    GOSUB *QUERY_WRITE_REG_L &H500+I*2 1    /* 0=incremental, 1=absolute
    GOSUB *QUERY_WRITE_REG_L &H580+I*2 0    /* 0=single, 1=join
  NEXT I
  GOSUB *QUERY_WRITE_REG_L &H400 300        /* position
  GOSUB *QUERY_WRITE_REG_L &H402 2700       /* position
  GOSUB *QUERY_WRITE_REG_L &H404 300        /* position
  GOSUB *QUERY_WRITE_REG_L &H406 1400       /* position
  GOSUB *QUERY_WRITE_REG_L &H408 2700       /* position
```

```
  GOSUB *QUERY_WRITE_REG_L &H40A 300            /* position
  GOSUB *QUERY_WRITE_REG_L &H40C 1400           /* position
  GOSUB *QUERY_WRITE_REG_L &H40E 2700           /* position

  GOSUB *QUERY_WRITE_REG_L &H1048 300           /* JOG pitch

RETURN

*SET_SPEED_PARAM

  GOSUB *QUERY_WRITE_REG_L &H284 0              /* Start Speed

  FOR I=0 TO 1
    GOSUB *QUERY_WRITE_REG_L &H480+I*2 MBK(1000)  /* speed
    GOSUB *QUERY_WRITE_REG_L &H600+I*2 MBK(1001)  /* acceleration
    GOSUB *QUERY_WRITE_REG_L &H680+I*2 MBK(1001)  /* deceleration
  NEXT I
  FOR I=2 TO 4
    GOSUB *QUERY_WRITE_REG_L &H480+I*2 MBK(1002)  /* speed
    GOSUB *QUERY_WRITE_REG_L &H600+I*2 MBK(1003)  /* acceleration
    GOSUB *QUERY_WRITE_REG_L &H680+I*2 MBK(1003)  /* deceleration
  NEXT I
  FOR I=5 TO 7
    GOSUB *QUERY_WRITE_REG_L &H480+I*2 MBK(1004)  /* speed
    GOSUB *QUERY_WRITE_REG_L &H600+I*2 MBK(1005)  /* acceleration
    GOSUB *QUERY_WRITE_REG_L &H680+I*2 MBK(1005)  /* deceleration
  NEXT I

  GOSUB *QUERY_WRITE_REG_L &H0286 MBK(1010)       /* JOG speed Hz
  GOSUB *QUERY_WRITE_REG_L &H0288 MBK(1011)       /* JOG acceleration/deceleration
  GOSUB *QUERY_WRITE_REG_L &H028A MBK(1012)       /* JOG start speed

RETURN

*HOME

  GOSUB *QUERY_WRITE_REG_L &H2C6 800            /* HOME Speed
  GOSUB *QUERY_WRITE_REG_L &H2C4 2000           /* HOME acceleration/deceleration
  GOSUB *QUERY_WRITE_REG_L &H21A 1              /* HOMES N.C.
  GOSUB *QUERY_WRITE_REG_L &H2CA 1              /* home dir

  ON -5                                         /* HOME
  GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)
  OFF -5
  GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)

  GOSUB *WAIT_FOR_READY
  GOSUB *READ_CURRENT_POS
  'GOSUB *P-PRESET

  OUT &H8 -1                                    /* [start | M2 M1 M0] -> [out -1]
  GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)
  OUT 0 -1
  GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)
  GOSUB *WAIT_FOR_READY_STOP

RETURN

*P-PRESET
  GOSUB *QUERY_WRITE_REG_L &H18A 1              /* P-PRESET
  GOSUB *QUERY_WRITE_REG_L &H18A 0
  GOSUB *WAIT_FOR_READY
  GOSUB *READ_CURRENT_POS
RETURN

*READ_CURRENT_POS                               /* current position read
  GOSUB *QUERY_READ_REG_L &HCC
  current_pos=(RES(4)<<24)+(RES(5)<<16)+(RES(6)<<8)+RES(7)
  PRINT "current_pos=" current_pos
RETURN

*WAIT_FOR_READY
  DO
    GOSUB *QUERY_READ_REG &H7F
    IF RES(5)&&H20<>0 THEN                      /* ready bit
```

```
      BREAK
    END_IF
  LOOP
RETURN

*WAIT_FOR_READY_STOP
  DO
    GOSUB *QUERY_READ_REG &H7F
    IF RES(5)&&H20<>0 THEN                          /* ready bit
      BREAK
    END_IF

    IF RES(5)&&H1==0 THEN                           /* NET-OUT0 = R4 = IN0
      ON -6                                         /* STOP
      GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)
      OFF -6                                        /* STOP
      GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)
      ARD_STOP=SYSCLK
      PRINT "stop" ARD_STOP
      BREAK
    END_IF

  LOOP
RETURN

*R0_ON
  ON -9                                             /* R0
  GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)
RETURN
*R0_OFF
  OFF -9                                            /* R0
  GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)
RETURN
*R1_ON
  ON -10                                            /* R0
  GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)
RETURN
*R1_OFF
  OFF -10                                           /* R0
  GOSUB *QUERY_WRITE_REG &H7D IN(-1~Wrd)
RETURN



'===============
' 4byte write
'===============
*QUERY_WRITE_REG_L
  _VAR REG_ADD WRITE_DATA
  'PRINT "== WRITE ==" HEX$(REG_ADD) HEX$(WRITE_DATA)

  QUERY(1)=slave_address                            /* slave address
  QUERY(2)=&H10                                     /* write to register
  QUERY(3)=REG_ADD/256                              /* begining to write H
  QUERY(4)=REG_ADD&&HFF                             /* begining to write L
  QUERY(5)=0                                        /* write count H
  QUERY(6)=2                                        /* write count L
  QUERY(7)=4                                        /* write count * 2
  QUERY(8)=(WRITE_DATA>>24)&&HFF                    /* write data HH
  QUERY(9)=(WRITE_DATA>>16)&&HFF                    /* write data HL
  QUERY(10)=(WRITE_DATA>>8)&&HFF                    /* write data LH
  QUERY(11)=WRITE_DATA&&HFF                         /* write data LL
  GOSUB *QUERY_SND 11
  GOSUB *RES_RCV 6

RETURN

'===============
' 2byte write
'===============
*QUERY_WRITE_REG
  _VAR REG_ADD WRITE_DATA
  'PRINT "== WRITE ==" HEX$(REG_ADD) HEX$(WRITE_DATA)

  QUERY(1)=slave_address                            /* slave address
```

```
  QUERY(2)=&H06                                 /* write to register
  QUERY(3)=REG_ADD/256                          /* begining to write H
  QUERY(4)=REG_ADD&&HFF                         /* begining to write L
  QUERY(5)=(WRITE_DATA>>8)&&HFF                 /* write data H
  QUERY(6)=WRITE_DATA&&HFF                      /* write data L
  GOSUB *QUERY_SND 6
  GOSUB *RES_RCV 6

RETURN

'===============
' 4byte read
'===============
*QUERY_READ_REG_L
  _VAR REG_ADD
  'PRINT "== READ ==" HEX$(REG_ADD)

  QUERY(1)=slave_address                        /* slave address
  QUERY(2)=&H03                                 /* read from register
  QUERY(3)=REG_ADD/256                          /* begining to read H
  QUERY(4)=REG_ADD&&HFF                         /* begining to read L
  QUERY(5)=&H00                                 /* read count H
  QUERY(6)=&H02                                 /* read count L
  GOSUB *QUERY_SND 6
  GOSUB *RES_RCV 7

RETURN

'===============
' 2byte read
'===============
*QUERY_READ_REG
  _VAR REG_ADD
  'PRINT "== READ ==" HEX$(REG_ADD)

  QUERY(1)=slave_address                        /* slave address
  QUERY(2)=&H03                                 /* read from register
  QUERY(3)=REG_ADD/256                          /* begining to read H
  QUERY(4)=REG_ADD&&HFF                         /* begining to read L
  QUERY(5)=&H00                                 /* read count H
  QUERY(6)=&H01                                 /* read count L

  GOSUB *QUERY_SND 6
  GOSUB *RES_RCV 5

RETURN

'===============
' send query
'===============
*QUERY_SND
  _VAR DATA_CNT
  'PRINT "SEND"

  DIMCPY QUERY(1) BUFF(1) DATA_CNT              /* QUERY()->BUFF()
  GOSUB *CRC_CALC
  QUERY(DATA_CNT+1)=CRC_L
  QUERY(DATA_CNT+2)=CRC_H

  SEND$=""                                      /* The each string size is 256bytes
  str_pt=SEND$                                  /* str_pt is the pointer of SEND$
  FOR array_num=1 TO DATA_CNT+2
    POKE QUERY(array_num) str_pt                /* binary data create
    INC str_pt
    'PRINT "QUERY(" array_num ")=" HEX$(QUERY(array_num))
  NEXT array_num

  PRINT# 1 STR_LEN|DATA_CNT+2 SEND$
  PRINT# 5 STR_LEN|DATA_CNT+2 SEND$             /* monitor for debug

RETURN
'===============
' read response
'===============
*RES_RCV
```

```
  _VAR DATA_CNT
  'PRINT "RCV"

  FOR array_num=1 TO DATA_CNT+2
    INPUT# 1 CHR_C|1 RES$
    PRINT# 5 STR_LEN|1 RES$                    /* monitor for debug
    RES(array_num)=ASC(RES$)
    'PRINT "RES(" array_num ")=" HEX$(RES(array_num))
  NEXT array_num

  DIMCPY RES(1) BUFF(1) DATA_CNT               /* RES()->BUFF()
  GOSUB *CRC_CALC
  'PRINT "CRC L H" HEX$(CRC_L) HEX$(CRC_H)

  IF RES(DATA_CNT+1)<>CRC_L THEN               /* CRC compare RES to CACL
    PRINT "CRC L ERROR"
    END
  END_IF
  IF RES(DATA_CNT+2)<>CRC_H THEN
    PRINT "CRC H ERROR"
    END
  END_IF


RETURN

'===============
' CRC16 calculate
'===============
*CRC_CALC

'== SAMPLE1 ==
  CRC=&HFFFF
  FOR crc_i=1 TO DATA_CNT
    crc_next=BUFF(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
```