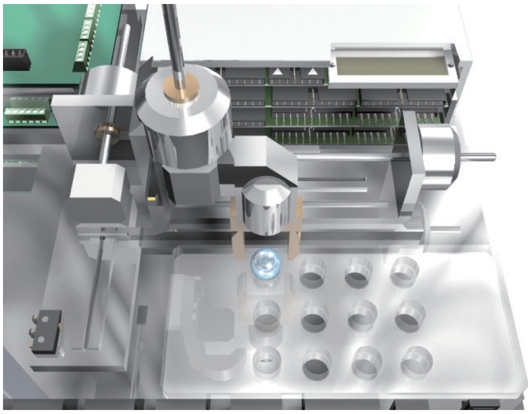
Chapter 7 Program Examples

7-1 Robot Applications

Here, a basic program description is illustrated using simple program examples.
Assumed is that glass balls arranged in the matrix of 4 x 3 shown below are grabbed and ejected at the origin.
This kind of application also requires a PG configuration and return to the origin.



```
CONST   HAND 15                       Definition of I/Os
 CONST   READY 0                      If symbols are assigned to I/Os using CONST in this way, making the
 CONST   BUSY 1                       program more readable.
 CONST   START 192
 PG   1                               PG selection.  PG of DSW1 is selected.
 ACCEL   X_A|Y_A|SACL 50000           Speed and acceleration are set.
 ACCEL   Z_A|SACL 40000               Configuration can be performed for each axis.  SACL specifies the S-curve.
 GOSUB   *HOME                        Subroutine call
 PALLET   1 P(1) P(2) P(3) P(4) 4 3   Pallet declaration.  Here, four points are specified.

 DO                                   Iteration control statement
  ON   READY
  QUIT   1 : TIME   10
  OFF   BUSY
  WAIT   M_SW(START)                  Wait for the start switch.  If there is chatter, M_SW.
  OFF   READY
  FORK   1 *BUSY                      Display BUSY (blinking) as a separate task.
  FOR   I=1 TO 12                     Sequential processing, works of 1~12 are extracted.
   JUMP   PL(1;I)
   WAIT   RR(ALL_A)==0                Monitoring the end of pulse generation (movement)
   ON   HAND                          Grab a glass ball.
   TIME   300
   JUMP   0 0 0 0                     Transport to the ejection position.
   OFF   HAND                         Release the glass ball.
   TIME   300
  NEXT                                End of sequential processing
 LOOP                                 End of iterations.
 END
*BUSY                                 Program of a separate task.
```

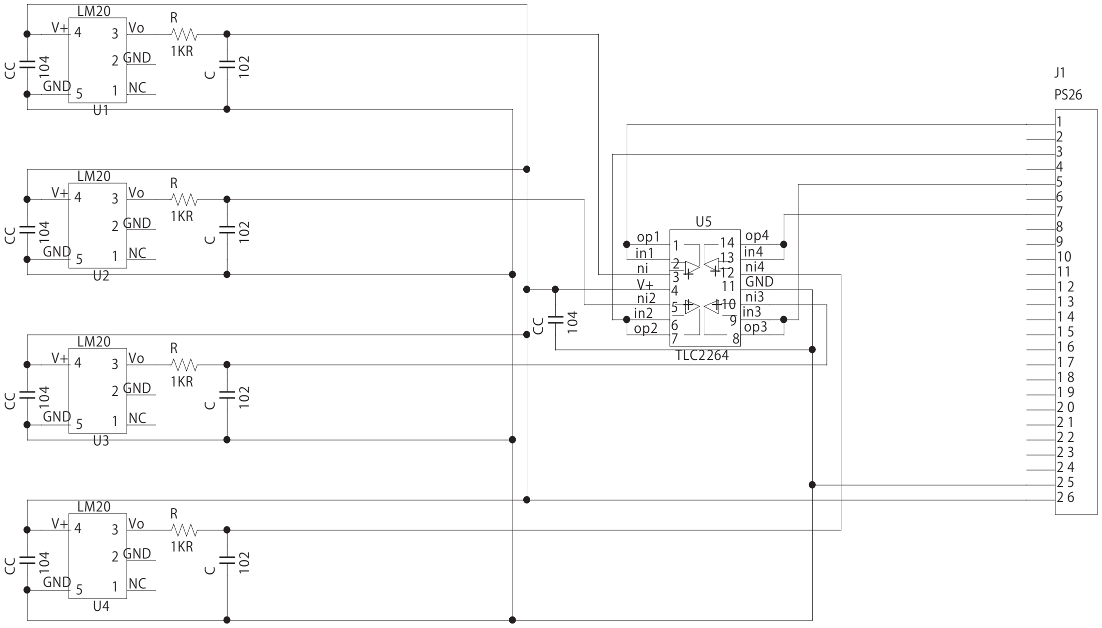
```
DO
  ON   BUSY                          Simple program which repeats LED blinking
  TIME   100
  OFF   BUSY
  TIME   100
 LOOP
*HOME                                Origin return program
 STOP   ALL_A VOID
 RMVS   1000 1000 0 -1000            Escape from the origin position
 WAIT   RR(ALL_A)==0
 SPEED   1000                        Set the speed to 1 kPPS
 STOP   ALL_A IN0_ON                 Specify a stopping condition.  (Origin sensor)
 RMVS   -100000 -100000 0 100000     Origin search
 WAIT   RR(ALL_A)==0                 End if stopped.
 STOP   ALL_A VOID                   Release the stopping condition.
 CLRPOS                              Set the current position to 0.
 SPEED   30000                       Restore the speed.
 RETURN                              Return from the subroutine.
```

Here, several conditions are set in order to make the program more readable.
The assumption is that, when turning the power on, XY should be near the origin,which, if stopped by an origin search, the position is always the origin. After exiting the
origin search, the sensor condition and the condition of others needs to be checked.

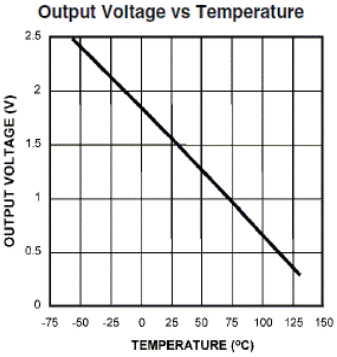
7-2 Temperature Measurement and Data Logger

By combining MPC-AD12 and the temperature sensor LM20, a multiple-point temperature measurement logger can be constructed. Below is the circuit diagram of the sensor. The distance between an OP amp and a sensor IC can be expanded by about 1 m. In the following circuit diagram, J1 connects to MPC-AD12, and the sensor power is supplied from MPC-AD12.
Illustrated here is a program which measures temperature data every 10 seconds in synchronization with the calendar IC and records/stores it in USB memory.

[LM20 reference circuit diagram]



[Method of conversion to temperature data]



LM20 has the characteristic shown at left. This is 1.86 V at 0 degree and changes by -11.7 mV every 1 degree.
MPC-AD12 has 1 mV / 1 digit as delivered. Therefore, 3-digit temperature data can be obtained by the following operation.

```
A=(1866-AD(0))*100/117
```

If A becomes 254, it signifies 25.4 degrees.

```
  DIM   DATA(4)
  TIME   500
 FILE$="log_S.txt"                         Specification of the log file of USB memory.
 err_cnt=0                                 (Eight characters .TXT in DOS format)
 err$="  "
*ondo
  FORK   1 *disp_lcd                       Commence task of displaying onto an LCD.
  ON_ERROR  *usb_err                       USB memory write error definition
  DO                                       Iteration
   WAIT   TIME(255)%16==0                  Wait for timing every 10 seconds.
   FOR   I=0 TO 3                          Iteration over 4 channels
 DATA(I)=(1866-AD(1,I))*100/117            Conversion of sensor voltage to temperature.
   NEXT
 DISP$=""
   FOR   I=0 TO 3                          Counting repetition over CH0~CH3
    FORMAT   " CH0_"
 DISP$=DISP$+STR$(I)
    FORMAT   "00.0"
 DISP$=DISP$+STR$(DATA(I))                 Conversion into character strings by each channel
   NEXT
   FORMAT   "000"                          Generation of character strings for LCD
 OND1$=STR$(DATA(0))                       Generation of date data character string.
 OND2$=STR$(DATA(1))
 I$=DATE$(3)+" "+TIME$(1)+DISP$+"\n"       Combine date, time, and temperature data.
   ON   768                                Write lamp lit.
   USB_WRITE   I$                          Write into USB memory.
   PRINT   I$                              Display in FTM.
   TIME   1000
   OFF   768                               Write lamp extinguished.
  LOOP
*usb_err                                   USB memory write error processing.
  RST_USB
  INC   err_cnt
  FORMAT   "00"
 err$=STR$(err_cnt)                        Conversion of error count to character string.
  PRINT   "ERRORUSB"
  TIME   500
  RESUME                                   Return to error generation command.
  END
*disp_lcd
  DO
 lcd$=err$+OND1$+OND2$                     Display number of errors and the temperature of CH0 and
   PR_LCD   lcd$                           CH1 on the LCD of an MPC.
   TIME   5000
  LOOP
```

A simple data logger can be thus constructed.
Data written into USB memory is as follows. If can be referred to using an ordinary PC.

```
"log_S.txt"
2009-05-29 10:36:30 CH0_22.2 CH1_21.5 CH2_21.6 CH3_22.0
2009-05-29 10:36:40 CH0_22.2 CH1_21.5 CH2_21.6 CH3_22.1
2009-05-29 10:36:50 CH0_22.2 CH1_21.5 CH2_21.6 CH3_22.1
```

Here, because the USB memory is a memory device containing flash memory, it can only withstand rewriting up to about 100 thousand times (some scatter occurs depending upon the manufacturer). If one rewrite is performed each second, 86,400 rewrites occur in 24 hours, degrading the memory.
In company tests, some were broken by about one week of continuous operation of writing at 1-second intervals.
Therefore, it is recommended that data to be written should be buffered in an MPC if possible to reduce the number of USB_WRITE executions.
In addition, USB memory is a device in which write errors may occur.
In order to prevent a halt in operations, it is recommended that appropriate error processing (ON_ERROR) be built in.

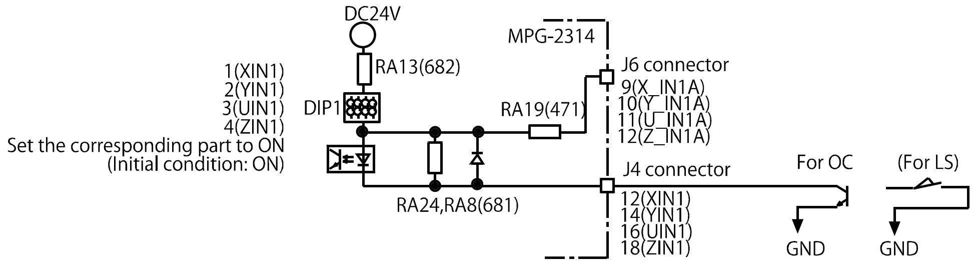
7-3 MPG-2314 Servo Driver Connection Examples

Servo drivers provide various kinds of input/output and have different connection methods, names, and the like.
Presented here are methods of connecting representative servo drivers manufactured by Yasukawa Electric and Panasonic with an MPG-2314, and illustrated are the corresponding origin return methods.

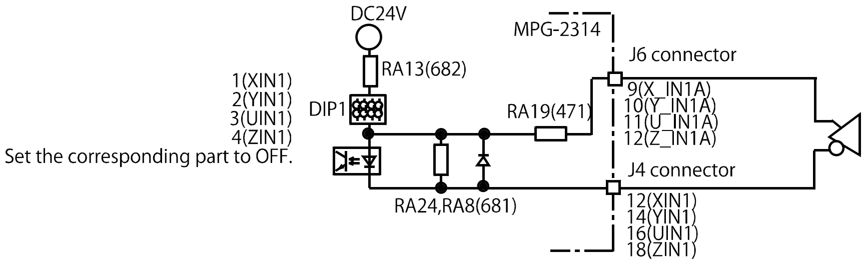
Origin input

For inputting the origin IN1 of each axis, an open collector output/contact or differential output can be connected.
In the case of the open collector output/contact, the DIP1 should be set to ON (default), and the J4 connector should be used for connection → (Fig. 1).
For differential input, the DIP1 should be set to OFF, and the J6 and J4 connectors should be used → (Fig. 2).
The IN0 input of each axis is only for an open collector output or contact → (Fig. 3).

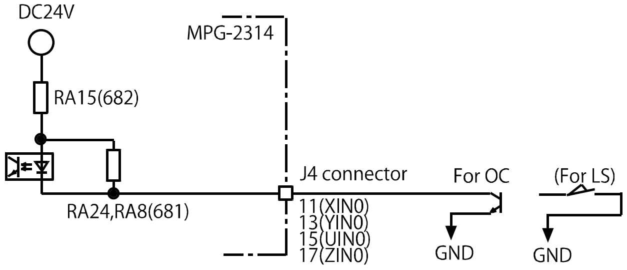
(Fig. 1) Connection with an open collector output/contact



(Fig. 2) Connection with a differential output

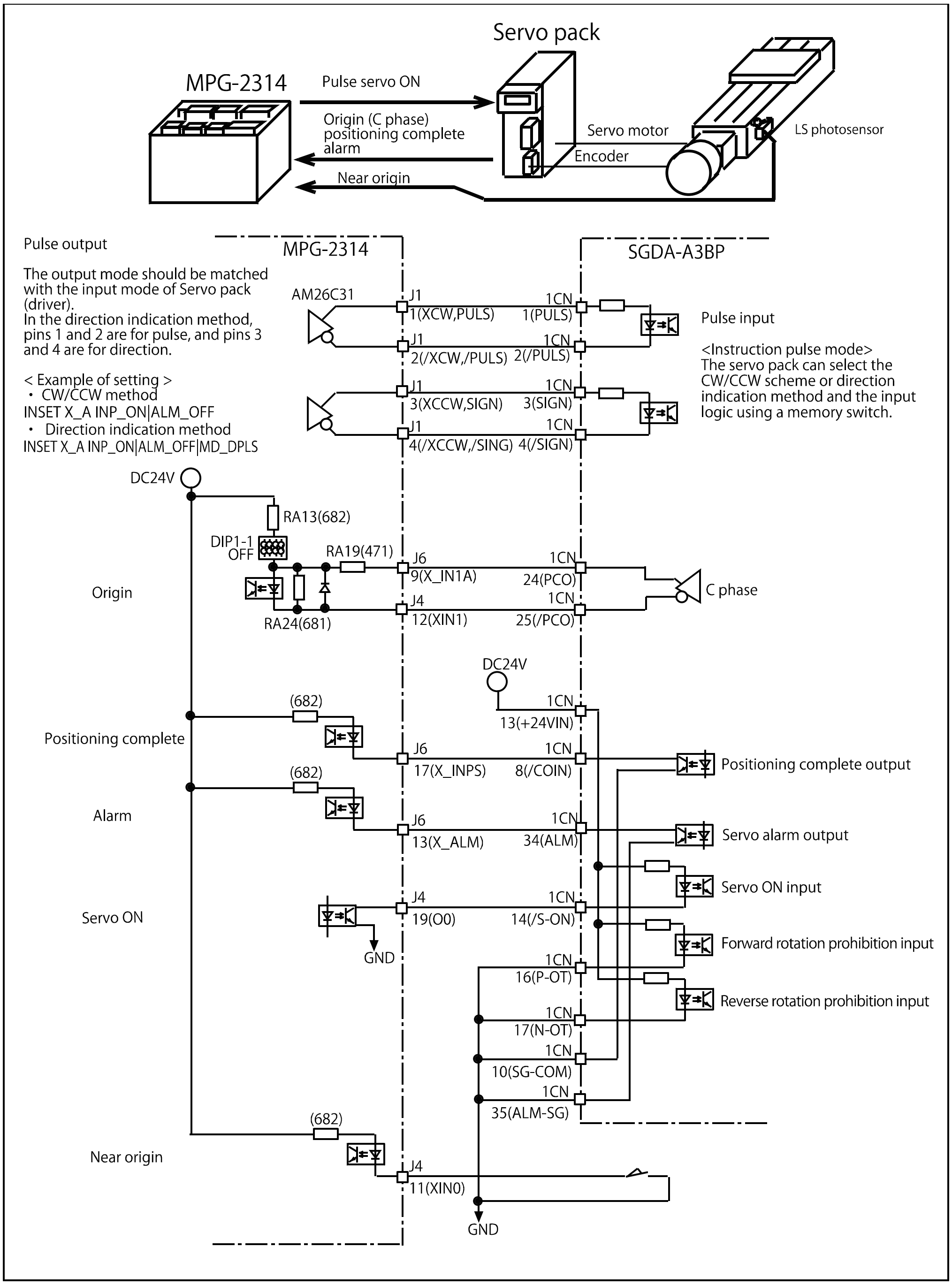


(Fig. 3) Connection of IN0



* RA24 and 8 are resistance arrays for a 2-line sensor.
* RA19 has an SIP socket installed. It can be replaced if necessary.

Connection example (Yasukawa Electric Corp. SGDA-A3BP)



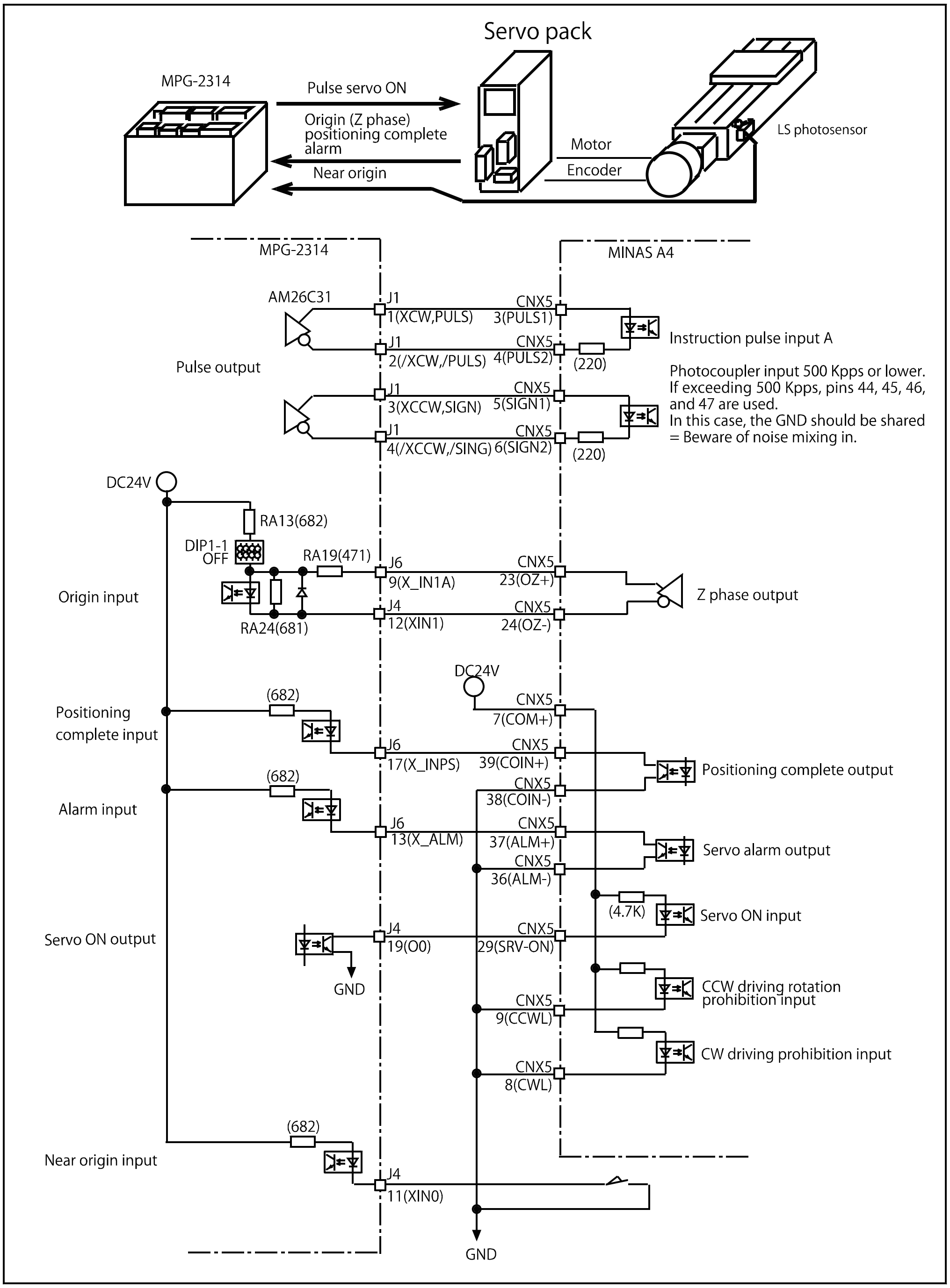
Pulse output

The output mode should be matched with the input mode of Servo pack (driver).
In the direction indication method, pins 1 and 2 are for pulse, and pins 3 and 4 are for direction.

< Example of setting >
・ CW/CCW method
INSET X_A INP_ON|ALM_OFF
・ Direction indication method
INSET X_A INP_ON|ALM_OFF|MD_DPLS

<Instruction pulse mode>
The servo pack can select the CW/CCW scheme or direction indication method and the input logic using a memory switch.

Connection example (Panasonic Corp. in the MINAS A4 positioning control mode)



Program example

```
/* Pressing down START SW → Servo ON → If near origin is ON, CW retreat movement →
/* Rotate somewhat fast in the CCW direction until near origin detection → Rotate slowly in the CCW
direction until C phase (Z phase) detection to clear coordinates
/* Afterwards, repeat pitch feed movement.
DO
  H_OFF 0                                /* Serve free
  PULSE_OUT 0 5                          /* START SW blinking
  WAIT SW(192)==0
  WAIT SW(192)==1                        /* Wait with START SW pressed down
  PULSE_OUT VOID
  ON 0
  PG 1                                   /* MPG-2314 DSW=1
/* Input and pulse mode set up: (a) is CW/CCW method, (b) is the direction indication method.
/* (a) Enabled when positioning complete = on | Enabled when servo alarm = off (Pulse output is CW/CCW method.)
  INSET X_A INP_ON|ALM_OFF
/* (b) enabled when positioning complete = on | Enabled when servo alarm = off | Pulse output =
direction indication method
  'INSET X_A INP_ON|ALM_OFF|MD_DPLS
  H_ON 0                                 /* Servo ON
  TIME 1000
  GOSUB *HOME_X
  ACCEL   X_A 50000 1000 1000
  FEED 100
  DO
    FOR I=1 TO 5
      RMVS X_A 10000
      WAIT RR(X_A)==0
      GOSUB *STOP_STATUS
      TIME 100
    NEXT I
    MOVS X_A 0
    WAIT RR(X_A)==0
    GOSUB *STOP_STATUS
    IF SW(192)==1 THEN
      BREAK
    END_IF
    TIME 1000
  LOOP
LOOP
*HOME_X                                  /* X-axis origin return
  ACCEL X_A 500 100 100                  /* Origin return speed
  FEED 100
  IF HPT(XIN0)!=0 THEN                   /* If X-axis IN0 is on, retreat movement.
    RMVS X_A 1000
    WAIT RR(X_A)==0
    TIME 100
  END_IF
  SHOM   X_A IN0_ON|IN1_ON|CCW           /* Near origin → Z phase detection in the CCW direction
  TMOUT 20000
  HOME -100000 0 0 0                     /* Near origin detection in the CCW direction
  IF timer_==0 THEN
    PRINT "TIME OUT"
    END
  END_IF
  STPS 0 VOID VOID VOID                  /*X set here to ‘0’.
  PRINT "X HOME" X(0)
  TIME 1000
  RETURN
*STOP_STATUS                             /* Stop status check
  IF RR(X_E)<>0 THEN                     /* Stop status judged from the error status
  /*IF LMT(X_A,ALM)==1 THEN              /* Stop status judged from the error input
    PRINT "Abnormal stop"
    PRX RR(X_E)
    END
  END_IF
  RETURN
```