## รายการอ้างอิง

<u>ภาษาไทย</u>

ไพศาล สงวนหมู่ และ ยืน ภู่วรวรรณ. การสื่อสารข้อมูลและไมโครคอมพิวเตอร์เนตเวอร์ค. พิมพ์ครั้งที่ 3. กรุงเทพมหานคร: บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2531.

สมศักดิ์ กีรติวุฒิเศรษฐ์. หลักการและการใช้งาน เครื่องมือวัดอุตสาหกรรม. กรุงเทพมหานคร: สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), 2528.

จิติ หนูแก้ว. เทคนิคการเชื่อมต่อ IBM PC กับอุปกรณ์ภายนอกเพื่อประยุกต์ใช้ในงานต่าง ๆ. กรุงเทพมหานคร: บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2535.

ปรเมษฐ์ ประนยานันทน์ และ ปิยพงศ์ เผ่าวนิช. คู่มือและการประยุกต์ใช้งาน ไมโครคอนโทรลเลอร์ MCS-51. กรุงเทพมหานคร: บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2536.

นิพนธ์ ศิริรัตน์. MCS-51 กับการติดต่อสื่อสารทางพอร์ตอนุกรม. วารสารเซมิคอนดักเตอร์อิเล็กทรอนิคส์ 120 (กันยายน 2535): 49-53.

________ ไมโครฯ แชนแนล ตอน MCS-51 กับ LCD. วารสารเซมิคอนดัคเตอร์อิเล็กทรอนิคส์ 121 ( ตุลาคม 2535 ): 117-121.

เอกชัย สันกำแพง. วงจรแปลง RS-232 เป็น RS-422. วารสารเซมิคอนดักเตอร์อิเล็กทรอนิคส์ 115 (มีนาคม-เมษายน 2535): 132-136.

ศิริชัย ดิลกรัตนพิจิตร และ ณรงค์ศักดิ์ ตั้งกาญจนศรี. อินพุต/เอาต์พุต อัตโนมัติ สำหรับงานอุตสาหกรรม ตอนที่ 1. วารสารเซมิคอนดักเตอร์อิเล็กทรอนิคส์ 115 (มีนาคม-เมษายน 2535): 20-29.

_______ อินพุต/เอาต์พุต อัตโนมัติ สำหรับงานอุตสาหกรรม ตอนจบ. วารสารเซมิคอนดักเตอร์อิเล็กทรอนิคส์ 116 (พฤษภาคม-มิถุนายน 2535): 19-22.

เปรมจิตร วิสุทธิศิริ. พื้นฐานวงจร เอทูดี, ดีทูเอ ตอน 2 วงจรแปลงอะนาล็อกเป็นดิจิตอล. วารสารเซมิคอนดักเตอร์อิเล็กทรอนิคส์ 103 ( ธ.ค. 2533 ): 302-309.

รณรงค์ ชนากร. ไมโครฯแชนแนล ตอน ต่อคีย์บอร์ดเข้ากับไมโครคอนโทรลเลอร์ 8052. วารสารเซมิคอนดักเตอร์อิเล็กทรอนิคส์ 136 ( กุมภาพันธ์ 2537 ): 57-61.

ทีมงานอีทีที. จออักษร LCD สำหรับซิงเกิ้ลบอร์ด. วารสารเซมิคอนดักเตอร์อิเล็กทรอนิคส์ 99 ( เมษายน-พฤศภาคม 2533 ): 231-240.

อุทัย จึงภักดี. แนวทางออกแบบเครื่องมือวัดทางอิเล็กทรอนิคส์ ตอนที่ 2, ทรานสดิวเซอร์ตรวจวัดอุณหภูมิ. วารสารเซมิคอนดักเตอร์อิเล็กทรอนิคส์ 138. ( เมษายน 2537 ): 115-121

ภาษาอังกฤษ

Tompkins, Willis J. Interfacing sensors to the IBM PC. Englewood cliffs N.J.: Prentice-Hall, 1992.

Hall, Douglas. Microprocessor and interfacing. New York.: McGraw-Hill, 1992.

Slater, Michael. Microprocessor-Based designed; A comprehensive to effective hardware design. Englewood cliffs N.J.: Prentice-Hall, 1992.

Krishna Kant. Microprocessor based Data acquisition system design.

New Delhi: Tata Mcgraw-Hill publishing Company Limited, 1987.

# ภาคผนวก ก.

## โปรแกรมภาษาแอสเซมบลี้ที่ใช้ควบคุมระบบแสวงหาข้อมูล

```
                ORG     0000H

COMMAND         EQU     0E060H   ; Read-Write Register
READBUSY        EQU     0E061H   ; Read BF(Busy Flag) and address
WRITEDATA       EQU     0E062H   ; Write character

KEYADDR         EQU     0E080H   ; 8255#3 Key board address

PORT_A1         EQU     0E000H
PORT_B1         EQU     0E001H
PORT_C1         EQU     0E002H
CTRL_1          EQU     0E003H

PORT_A2         EQU     0E004H
PORT_B2         EQU     0E005H
PORT_C2         EQU     0E006H
CTRL_2          EQU     0E007H

PORT_A3         EQU     0E008H
PORT_B3         EQU     0E009H
PORT_C3         EQU     0E00AH
CTRL_3          EQU     0E00BH

PORT_A4         EQU     0E00CH
PORT_B4         EQU     0E00DH
```

```
PORT_C4         EQU     0E00EH
CTRL_4          EQU     0E00FH
AIN_A           EQU     0E010H
AIN_B           EQU     0E011H
AIN_C           EQU     0E012H
CTRL            EQU     0E013H

RTC             EQU     0E0E0H
SEC1            EQU     RTC
SEC10           EQU     RTC+1
MIN1            EQU     RTC+2
MIN10           EQU     RTC+3
HOUR1           EQU     RTC+4
HOUR10          EQU     RTC+5
DAY1            EQU     RTC+6
DAY10           EQU     RTC+7
MONTH1          EQU     RTC+8
MONTH10         EQU     RTC+9
YEAR1           EQU     RTC+0AH
YEAR10          EQU     RTC+0BH
CREG_D          EQU     RTC+0DH
CREG_E          EQU     RTC+0EH
CREG_F          EQU     RTC+0FH
                JMP     INITIAL

                ORG     0003H
                JMP     RTC_INT

                ORG     0013H           ; external interupt servive routine
                JMP     KEY_INT
                ORG     0023H
                CLR     ES
                MOV     A,SBUF
                CLR     RI
                LCALL   CHECK
                SETB    ES
                RETI

INITIAL:        SETB    EA
                SETB    IT0             ; set INT0 mode
                SETB    EX0
                SETB    IT1             ; set INT1 mode
                SETB    EX1
                MOV     SCON,#50H
                MOV     TMOD,#20H
                MOV     TH1,#0FDH
                SETB    TR1
                LCALL   RTC_INIT
                LCALL   LCD_INIT
                MOV     R0,#82H         ; 8255 ctrl code
                MOV     @R0,#90H
                INC     R0
                MOV     @R0,#92H
                INC     R0
                MOV     @R0,#9BH
                MOV     R0,#A0H         ; clear analog input maximum buffer
                MOV     @R0,#00
                MOV     R0,#A1H
                MOV     @R0,#00
                MOV     R0,#A2H
                MOV     @R0,#00
                MOV     R0,#A3H
                MOV     @R0,#00
                MOV     R0,#A4H
                MOV     @R0,#00
                MOV     R0,#A5H
                MOV     @R0,#00
                MOV     R0,#A6H
                MOV     @R0,#00
                MOV     R0,#A7H
                MOV     @R0,#00
                MOV     R0,#A8H
                MOV     @R0,#00
MAIN:           SETB    ES
                JMP     MAIN

KEY_INT:        CLR     EA
                MOV     DPTR,#KEYADDR
                MOVX    A,@DPTR
                ANL     A,#0FH
                MOV     R5,A
                CLR     IE1
                SETB    EA
                RETI
RTC_INT:        PUSH    ACC
                MOV     DPTR,#CREG_D
                MOVX    A,@DPTR
                CLR     ACC.2
                MOVX    @DPTR,A
                LCALL   READTIME
                POP     ACC
                RETI
```

```
;*****************************;
; CHECK COMMAND SUBRUTIEN ;
;*****************************;
CHECK:          CJNE     A,#41H,CHECKB
                LCALL    COMMANDA
                RET
CHECKB:         CJNE     A,#42H,CHECKI
                LCALL    COMMANDB
                RET
CHECKI:         CJNE     A,#49H,CHECKU
                LCALL    SETUP
                LCALL    PORT_INIT
                RET
CHECKU:         CJNE     A,#55H,CHECKAA
                LCALL    COMMANDU
                RET
CHECKAA:        CJNE     A,#61H,CHECKBB
                LCALL    COMMAN_A
                RET
CHECKBB:        CJNE     A,#62H,CHECKEE
                LCALL    COMMAN_B
                RET
CHECKEE:        CJNE     A,#65H,CHECKFF
                LCALL    COMMANDE
                RET
CHECKFF:        CJNE     A,#66H,CHECKUU
                LCALL    COMMANDF
                RET
CHECKUU:        CJNE     A,#75H,CHECKVV
                LCALL    COMMAN_U
```

```
            RET
CHECKVV:    CJNE    A,#56H,CHECKR
            LCALL   COMMANDV
            RET
CHECKR:     CJNE    A,#52H,CHECKL
            LCALL   RESET
            RET
CHECKL:     CJNE    A,#4CH,CHECKN
            RET
CHECKN:     CJNE    A,#4EH,CHECKC
            RET
CHECKC:     CJNE    A,#43H,CHECKD
            LCALL   COMMANDC
            RET
CHECKD:     CJNE    A,#44H,QUIT
            LCALL   COMMANDD
QUIT:       RET

READTIME:   MOV     DPTR,#SEC1
            LCALL   RTC_READ
            MOV     DPTR,#BUFFER
            MOVX    @DPTR,A
            MOV     DPTR,#MIN1
            LCALL   RTC_READ
            MOV     DPTR,#BUFFER+1
            MOVX    @DPTR,A

            MOV     DPTR,#HOUR1
            LCALL   RTC_READ
            ANL     A,#00111111B
            MOV     DPTR,#BUFFER+2
            MOVX    @DPTR,A

            MOV     DPTR,#DAY1
            LCALL   RTC_READ
            MOV     DPTR,#BUFFER+3
            MOVX    @DPTR,A

            MOV     DPTR,#MONTH1
            LCALL   RTC_READ
            MOV     DPTR,#BUFFER+4
            MOVX    @DPTR,A

            MOV     DPTR,#YEAR1
            LCALL   RTC_READ
            MOV     DPTR,#BUFFER+5
            MOVX    @DPTR,A
            RET
RTC_READ:   MOVX    A,@DPTR
            ANL     A,#0FH
            MOV     R4,A
            INC     DPTR
            MOVX    A,@DPTR
            LCALL   SHIFT
            ANL     A,#F0H
            ADD     A,R4
            RET

SHIFT:      RL      A
            RL      A
            RL      A
            RL      A
```

```
                RET

CHK_KEY:        MOV     DPTR,#TABLE
                MOV     A,R5
                RL      A
                JMP     @A+DPTR
TABLE:          AJMP    RETURN
                AJMP    CASE1
                AJMP    CASE2
                AJMP    CASE3
                AJMP    RETURN
                AJMP    CASE5
                AJMP    CASE6
                AJMP    CASE7
                AJMP    RETURN
                AJMP    CASE9
                AJMP    CASE10
                AJMP    CASE11
                AJMP    RETURN
                AJMP    CASE13
                AJMP    CASE14
                AJMP    CASE15
CASE1:          MOV     A,#3
                JMP     RETURN
CASE2:          MOV     A,#2
                JMP     RETURN
CASE3:          MOV     A,#1
                JMP     RETURN
CASE5:          MOV     A,#6
                JMP     RETURN
CASE6:          MOV     A,#5
```

```
                JMP     RETURN
CASE7:          MOV     A,#4
                JMP     RETURN
CASE9:          MOV     A,#9
                JMP     RETURN
CASE10:         MOV     A,#8
                JMP     RETURN
CASE11:         MOV     A,#7
                JMP     RETURN
CASE14:         MOV     A,#0
                JMP     RETURN
CASE13:         JMP     RETURN
CASE15:         JMP     RETURN
RETURN:         RET

;*****************************
;           SET UP
;*****************************
SETUP:          LCALL   CLR_SCR
                SETB    TI
                MOV     DPTR,#LINEI1
                MOV     R1,#12
STRI1:          CLR     A
                MOVC    A,@A+DPTR
                LCALL   WRITE
                INC     DPTR
                DJNZ    R1,STRI1
                LCALL   CLR_SCR
AIN:            MOV     DPTR,#LINEI2
                MOV     R1,#11
STRI2:          CLR     A
```

```
            MOVC    A,@A+DPTR
            LCALL   WRITE
            INC     DPTR
            DJNZ    R1,STRI2
            MOV     A,#0DH
            LCALL   TRANSMIT
            LCALL   MRECV
            MOV     R1,#3FH
            MOV     @R1,#2CH
            LCALL   PARA2
            MOV     R0,#70H
            MOV     @R0,A
            LCALL   HEXTODEC
            MOV     A,52H
            LCALL   WRITE
            MOV     A,53H
            LCALL   WRITE
AOUT:       MOV     A,#40H
            LCALL   GOTOXY
            MOV     DPTR,#LINEI3
            MOV     R1,#11
STRI3:      CLR     A
            MOVC    A,@A+DPTR
            LCALL   WRITE
            INC     DPTR
            DJNZ    R1,STRI3
            MOV     A,#0DH
            LCALL   TRANSMIT
            LCALL   MRECV
            MOV     R1,#3FH
            MOV     @R1,#2CH
```

```
                LCALL   PARA2
                MOV     R0,#71H
                MOV     @R0,A
                LCALL   HEXTODEC
                MOV     A,52H
                LCALL   WRITE
                MOV     A,53H
                LCALL   WRITE
DIN:            MOV     A,#10H
                LCALL   GOTOXY
                MOV     DPTR,#LINEI4
                MOV     R1,#12
STRI4:          CLR     A
                MOVC    A,@A+DPTR
                LCALL   WRITE
                INC     DPTR
                DJNZ    R1,STRI4
                MOV     A,#0DH
                LCALL   TRANSMIT
                LCALL   MRECV
                MOV     R1,#3FH
                MOV     @R1,#2CH
                LCALL   PARA2
                MOV     R0,#72H
                MOV     @R0,A
                LCALL   HEXTODEC
                MOV     A,52H
                LCALL   WRITE
                MOV     A,53H
                LCALL   WRITE
DOUT:           MOV     A,#50H
```

```
                LCALL   GOTOXY
                MOV     DPTR,#LINEI5
                MOV     R1,#12
STRI5:          CLR     A
                MOVC    A,@A+DPTR
                LCALL   WRITE
                INC     DPTR
                DJNZ    R1,STRI5
                MOV     A,#0DH
                LCALL   TRANSMIT
                LCALL   MRECV
                MOV     R1,#3FH
                MOV     @R1,#2CH
                LCALL   PARA2
                MOV     R0,#73H
                MOV     @R0,A
                LCALL   HEXTODEC
                MOV     A,52H
                LCALL   WRITE
                MOV     A,53H
                LCALL   WRITE
                MOV     R2,#00
                MOV     R4,#B0H
LIM_LOOP:       MOV     A,#0DH
                LCALL   TRANSMIT
                LCALL   MRECV
                MOV     R1,#3FH
                MOV     @R1,#2CH
                LCALL   PARA2
                MOV     R3,A
                MOV     A,R4
```

```
            MOV     R0,A
            MOV     A,R3
            MOV     @R0,A
            INC     R4
            INC     R2
            MOV     A,R2
            CJNE    A,70H,LIM_LOOP
            LCALL   DELAY_1MS
            MOV     A,#0DH
            LCALL   TRANSMIT
            MOV     R2,#00
            MOV     R4,#E0H
ANA_TYPE:   MOV     A,#0DH
            LCALL   TRANSMIT
            LCALL   MRECV
            MOV     R1,#3FH
            MOV     @R1,#2CH
            LCALL   PARA2
            MOV     R3,A
            MOV     A,R4
            MOV     R0,A
            MOV     A,R3
            MOV     @R0,A
            INC     R4
            INC     R2
            MOV     A,R2
            CJNE    A,70H,ANA_TYPE
            MOV     A,#0DH
            LCALL   TRANSMIT
            CLR     TI
END_SETUP:  RET
```

```
;***************************************
;           COMMAND B
;     WRITE BYTE TO BLOCK OF OUTPUT LINES
;     PARAMETER1 <0 TO 12>   IN 80H
;     PARAMETER2 <0 OR 255> IN 81H
;     DIGITAL I/O DATA  IN C0H-CBH
;           USE REG A,R0,R1
;***************************************
COMMANDB:   LCALL   CLR_SCR
            SETB    TI
            LCALL   MRECV
            LCALL   PARA1
            MOV     R1,#80H          ; card to be write store to 80H
            MOV     @R1,A

            LCALL   PARA2
            MOV     R1,#81H
            MOV     @R1,A            ; store value to write in 81H
            MOV     A,#C0H
            MOV     R0,#80H
            ADD     A,@R0
            MOV     R0,A             ; get data output address

            MOV     R1,#80H
            MOV     A,@R1            ; load card no. to ACC
            CJNE    A,#0,B_PORT_1
            MOV     DPTR,#PORT_A1
            JMP     WRITE1
B_PORT_1:   CJNE    A,#1,B_PORT_2
            MOV     DPTR,#PORT_B1
            JMP     WRITE1
```

```
B_PORT_2:       CJNE    A,#2,B_PORT_3
                MOV     DPTR,#PORT_C1
                JMP     WRITE1
B_PORT_3:       CJNE    A,#3,B_PORT_4
                MOV     DPTR,#PORT_A2
                JMP     WRITE1
B_PORT_4:       CJNE    A,#4,B_PORT_5
                MOV     DPTR,#PORT_B2
                JMP     WRITE1
B_PORT_5:       CJNE    A,#5,B_PORT_6
                MOV     DPTR,#PORT_C2
                JMP     WRITE1
B_STOP2:        JMP     B_STOP
B_PORT_6:       CJNE    A,#6,B_PORT_7
                MOV     DPTR,#PORT_A3
                JMP     WRITE1
B_PORT_7:       CJNE    A,#7,B_PORT_8
                MOV     DPTR,#PORT_B3
                JMP     WRITE1
B_PORT_8:       CJNE    A,#8,B_PORT_9
                MOV     DPTR,#PORT_C3
                JMP     WRITE1
B_PORT_9:       CJNE    A,#9,B_PORT_10
                MOV     DPTR,#PORT_A4
                JMP     WRITE1
B_PORT_10:      CJNE    A,#10,B_PORT_11
                MOV     DPTR,#PORT_B4
                JMP     WRITE1
B_PORT_11:      MOV     DPTR,#PORT_C4
```

```
WRITE1:         INC     R1
                MOV     A,@R1           ; load data to be written to ACC
                MOVX    @DPTR,A
                MOV     @R0,A            ; save data output
                LCALL   CLR_SCR
                MOV     DPTR,#LINEB1
                MOV     R1,#12
STRB1:          CLR     A
                MOVC    A,@A+DPTR
                LCALL   WRITE
                INC     DPTR
                DJNZ    R1,STRB1

                MOV     R0,#81H
                LCALL   HEXTODEC
                LCALL   LCD_3_CH

                MOV     A,#40H
                LCALL   GOTOXY
                MOV     DPTR,#LINEB2
                MOV     R1,#14
STRB2:          CLR     A
                MOVC    A,@A+DPTR
                LCALL   WRITE
                INC     DPTR
                DJNZ    R1,STRB2

                MOV     A,#10H
                LCALL   GOTOXY
                MOV     DPTR,#LINEB3
                MOV     R1,#8
```

```
STRB3:          CLR     A
                MOVC    A,@A+DPTR
                LCALL   WRITE
                INC     DPTR
                DJNZ    R1,STRB3

                MOV     R0,#80H
                LCALL   HEXTODEC
                CLR     A
                MOV     A,53H
                LCALL   WRITE
                MOV     A,#0DH
                LCALL   TRANSMIT
B_STOP:         CLR     TI
                RET

;*****************************************
;               COMMAND C
;         ALL DIGITAL OUTPUT TO 0
;       DIGITAL I/O ADDRESS C0H-CBH
;               USE REG R0
;*****************************************
COMMANDC:       LCALL   CLR_SCR
                SETB    TI

                MOV     A,#00H
                MOV     DPTR,#PORT_A1
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_B1
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_C1
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_A2
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_B2
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_C2
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_A3
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_B3
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_C3
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_A4
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_B4
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_C4
                MOVX    @DPTR,A

CLR_MEM:        MOV     A,#C0H
                MOV     R0,#72H
                ADD     A,@R0
                MOV     R0,A            ; R0 = start output address
                MOV     R2,#00
CLM_LOOP:       MOV     @R0,#00H
                INC     R0
                INC     R2
                MOV     A,R2
                CJNE    A,73H,CLM_LOOP
```

```
            MOV     DPTR,#LINEC1
            MOV     R1,#37
STRC1:      CJNE    R1,#23,C_LINE2
            MOV     A,#40H
            LCALL   GOTOXY
            JMP     WRITEC
C_LINE2:    CJNE    R1,#10,WRITEC
            MOV     A,#10H
            LCALL   GOTOXY
WRITEC:     CLR     A
            MOVC    A,@A+DPTR
            LCALL   WRITE
            LCALL   TRANSMIT
            INC     DPTR
            DJNZ    R1,STRC1
            JMP     CLR_END

CLR_END1:   MOV     DPTR,#LINEC2
            MOV     R1,#32
STRC_1:     CJNE    R1,#22,LINE2_C
            MOV     A,#40H
            LCALL   GOTOXY
            JMP     WRITE_C
LINE2_C:    CJNE    R1,#10,WRITE_C
            MOV     A,#10H
            LCALL   GOTOXY
WRITE_C:    CLR     A
            MOVC    A,@A+DPTR
            LCALL   WRITE
            LCALL   TRANSMIT
            INC     DPTR
```

```
                DJNZ    R1,STRC_1

CLR_END:        MOV     A,#0DH
                LCALL   TRANSMIT
                CLR     TI
                RET

;*****************************************
;               COMMAND D
;         ALL DIGITAL OUTPUT TO 1
;       DIGITAL I/O ADDRESS C0H-CBH
;               USE REG R0
;*****************************************
COMMANDD:       LCALL   CLR_SCR
                SETB    TI

                MOV     A,#0FFH
                MOV     DPTR,#PORT_A1
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_B1
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_C1
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_A2
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_B2
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_C2
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_A3
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_B3
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_C3
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_A4
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_B4
                MOVX    @DPTR,A
                MOV     DPTR,#PORT_C4
                MOVX    @DPTR,A

SET_MEM:        MOV     A,#C0H
                MOV     R0,#72H
                ADD     A,@R0
                MOV     R0,A            ; R0 = start output address
                MOV     R2,#00
SM_LOOP:        MOV     @R0,#0FFH
                INC     R0
                INC     R2
                MOV     A,R2
                CJNE    A,73H,SM_LOOP

                MOV     DPTR,#LINED1
                MOV     R1,#37
STRD1:          CJNE    R1,#23,D_LINE2
                MOV     A,#40H
                LCALL   GOTOXY
                JMP     WRITED
D_LINE2:        CJNE    R1,#10,WRITED
                MOV     A,#10H
                LCALL   GOTOXY
```

```
WRITED:        CLR     A
               MOVC    A,@A+DPTR
               LCALL   WRITE
               LCALL   TRANSMIT
               INC     DPTR
               DJNZ    R1,STRD1
               JMP     SET_END

SET_END1:      MOV     DPTR,#LINEC2
               MOV     R1,#32
STRD_1:        CJNE    R1,#22,LINE2_D
               MOV     A,#40H
               LCALL   GOTOXY
               JMP     WRITE_D
LINE2_D:       CJNE    R1,#10,WRITE_D
               MOV     A,#10H
               LCALL   GOTOXY
WRITE_D:       CLR     A
               MOVC    A,@A+DPTR
               LCALL   WRITE
               LCALL   TRANSMIT
               INC     DPTR
               DJNZ    R1,STRD_1

SET_END:       MOV     A,#0DH
               LCALL   TRANSMIT
               CLR     TI
               RET
```

```
;*********************************************
;               COMMAND f
;     READ DIGITAL INPUT CARD 0 - 11
;     PARAMETER1 <0 TO 11>  IN 80h
;       ANSWER   <0 OR 255> IN C0-CHh and 55h
;              USE REG A,R0,R1,R2
;*********************************************
COMMANDF:     LCALL   CLR_SCR          ; execute by PC command
              LCALL   MRECV
              MOV     R1,#3FH
              MOV     @R1,#2CH
              LCALL   PARA2
              MOV     R0,#80H
              MOV     @R0,A

              MOV     R1,#80H
              MOV     A,@R1            ; load card no.0 - 11 to ACC
              CJNE    A,#0,F_PORT_1
              MOV     DPTR,#PORT_A1
              JMP     FSEND2
F_PORT_1:     CJNE    A,#1,F_PORT_2
              MOV     DPTR,#PORT_B1
              JMP     FSEND2
F_PORT_2:     CJNE    A,#2,F_PORT_3
              MOV     DPTR,#PORT_C1
              JMP     FSEND2
F_PORT_3:     CJNE    A,#3,F_PORT_4
              MOV     DPTR,#PORT_A2
              JMP     FSEND2
F_PORT_4:     CJNE    A,#4,F_PORT_5
              MOV     DPTR,#PORT_B2
```

```
                JMP     FSEND2
F_PORT_5:       CJNE    A,#5,F_PORT_6
                MOV     DPTR,#PORT_C2
                JMP     FSEND2
F_STOP2:        JMP     B_STOP
F_PORT_6:       CJNE    A,#6,F_PORT_7
                MOV     DPTR,#PORT_A3
                JMP     FSEND2
F_PORT_7:       CJNE    A,#7,F_PORT_8
                MOV     DPTR,#PORT_B3
                JMP     FSEND2
F_PORT_8:       CJNE    A,#8,F_PORT_9
                MOV     DPTR,#PORT_C3
                JMP     FSEND2
F_PORT_9:       CJNE    A,#9,F_PORT_10
                MOV     DPTR,#PORT_A4
                JMP     FSEND2
F_PORT_10:      CJNE    A,#10,F_PORT_11
                MOV     DPTR,#PORT_B4
                JMP     FSEND2
F_PORT_11:      MOV     DPTR,#PORT_C4

FSEND2:         MOVX    A,@DPTR
                MOV     R0,#50H
                MOV     @R0,A

                LCALL   CLR_SCR
                MOV     DPTR,#LINEF1
                MOV     R2,#14
STRF1:          CLR     A
                MOVC    A,@A+DPTR
```

```
            LCALL   WRITE
            INC     DPTR
            DJNZ    R2,STRF1

            MOV     A,#40H
            LCALL   GOTOXY
            MOV     DPTR,#LINEF2
            MOV     R2,#9
STRF2:      CLR     A
            MOVC    A,@A+DPTR
            LCALL   WRITE
            INC     DPTR
            DJNZ    R2,STRF2

            MOV     R0,#80H
            LCALL   HEXTODEC
            CLR     A
            MOV     A,53H
            LCALL   WRITE

            MOV     A,#10H
            LCALL   GOTOXY
            MOV     DPTR,#LINEF3
            MOV     R2,#12
STRF3:      CLR     A
            MOVC    A,@A+DPTR
            LCALL   WRITE
            INC     DPTR
            DJNZ    R2,STRF3
```

```
                MOV     A,#0DH
                LCALL   TRANSMIT
                MOV     R0,#50H
                LCALL   HEXTODEC
                LCALL   WRITE_3_CH
F_STOP:         CLR     TI
                RET

;*****************************************
;               COMMAND R
;             INITIALIZATION
;       DATA OUTPUT BLOCK0   IN 32H
;       DATA OUTPUT BLOCK1   IN 33H
;       DATA OUTPUT BLOCK2   IN 34H
;       DATA OUTPUT BLOCK3   IN 35H
;           USE DATA ADDRESS 20H
;               USE REG R0
;*****************************************
RESET:          LCALL   CLR_SCR
                SETB    TI

                MOV     R2,#00
                MOV     DPTR,#PORT_A1
                MOV     A,73H
                JZ      NO_OUTPUT
RESET_LOOP:     CJNE    R2,#12,D_O_1
                JMP     END_O_PORT
D_O_1:          INC     R2
                MOV     A,R2
                MOV     B,#4
                DIV     AB
```

```
            MOV     A,B
            CJNE    A,#0,R_CONTI
            INC     DPTR
R_CONTI:    MOV     A,#00H
            MOVX    @DPTR,A
            INC     DPTR
            JMP     RESET_LOOP

END_O_PORT: MOV     A,#32H
            MOV     R0,#72H
            ADD     A,@R0
            MOV     R0,A
            MOV     R2,#00
RM_LOOP:    MOV     @R0,#00H
            INC     R0
            INC     R2
            MOV     A,R2
            CJNE    A,73H,RM_LOOP

NO_OUTPUT:  MOV     R1,#68H
            MOV     @R1,#30H
            INC     R1
            MOV     @R1,#30H
            MOV     R1,#6EH
            MOV     @R1,#30H
            INC     R1
            MOV     @R1,#30H
            MOV     R1,#74H
            MOV     @R1,#30H
            INC     R1
            MOV     @R1,#30H
```

```
                MOV     R1,#7AH
                MOV     @R1,#30H
                INC     R1
                MOV     @R1,#30H
                MOV     2FH,#02H
                MOV     2EH,#02H
                MOV     2DH,#02H
                MOV     2CH,#02H
                MOV     A,#00
                MOV     R0,#90H
R_A_VAL:        MOV     @R0,#00         ; reset analog input value
                INC     R0
                INC     A
                CJNE    A,70H,R_A_VAL

                MOV     A,#00
                MOV     R0,#A0H
R_A_MAX:        MOV     @R0,#00         ; reset analog input maximum value
                INC     R0
                INC     A
                CJNE    A,70H,R_A_MAX

                MOV     A,#00
                MOV     R0,#B0H
R_A_LIM:        MOV     @R0,#255        ; reset analog input limit
                INC     R0
                INC     A
                CJNE    A,70H,R_A_LIM

                MOV     A,#40H
                LCALL   GOTOXY
                MOV     DPTR,#LINER1
                MOV     R0,#16
DISP_R1:        CLR     A
                MOVC    A,@A+DPTR
                LCALL   WRITE
                LCALL   TRANSMIT
                INC     DPTR
                DJNZ    R0,DISP_R1
                MOV     A,#0DH
                LCALL   TRANSMIT
                CLR     TI
                RET
;******************************************
;           COMMAND U
;   WRITE VOLTAGE TO ANALOG OUTPUT
;       PARAMETER1 <0 - 3>
;       PARAMETER2 <0 - 10.20>
;******************************************
COMMANDU:       LCALL   CLR_SCR
                SETB    TI
                LCALL   RECV            ; get analog output card no.
                MOV     R0,#80H
                MOV     @R0,A

                MOV     R1,#11H         ; used address 11H - 14H
                MOV     R4,#04H
XYZ:            MOV     @R1,#00H
                INC     R1
                DJNZ    R4,XYZ
                MOV     A,80H
                SUBB    A,#30H
                MOV     29H,A
                LCALL   BLOCK           ; save start address of desired
                MOV     R0,#81H         ; card in 81H
                MOV     A,R1
                MOV     @R0,A

                LCALL   RECV
                MOV     R0,#11H         ; address 11H store the first
                MOV     R2,#00H         ; value to be write to analog
                MOV     R3,#00H         ; output  (Ex 10.2v)
AAA:            LCALL   RECV            ; 11H -> 1H  (char 1 -> 56H)
                INC     R3              ; 12H -> 0H  (char 0 -> 57H)
                MOV     @R1,A           ;            (char . -> 58H)
                INC     R1              ; 13H -> 2H  (char 2 -> 59H)
                CJNE    A,#2EH,NOTDOT   ; 28H store character lengths
                JMP     AAA
NOTDOT:         CJNE    A,#0DH,NOTCR
                MOV     28H,R3
                JMP     ENDU
NOTCR:          SUBB    A,#30H
                MOV     @R0,A
                INC     R2              ; R2 store value lengths
                INC     R0
                JMP     AAA
STOP_U1:        JMP     STOP_U2
                MOV     B,R0            ; save content in reg R0 to B

ENDU:           MOV     DPTR,#LINEU1
                MOV     R0,#25
U_LINE1:        CJNE    R0,#8,U_DISP1
                MOV     A,#40H
                LCALL   GOTOXY
U_DISP1:        CLR     A
                MOVC    A,@A+DPTR
                LCALL   WRITE
                LCALL   TRANSMIT
                INC     DPTR
                DJNZ    R0,U_LINE1
                MOV     R0,#80H          ; get analog output card no.
                MOV     A,@R0
                LCALL   WRITE
                LCALL   TRANSMIT
                MOV     A,#' '
                LCALL   TRANSMIT
                MOV     A,#10H
                LCALL   GOTOXY
                MOV     DPTR,#LINEU2
                MOV     R0,#8
U_DISP2:        CLR     A
                MOVC    A,@A+DPTR
                LCALL   WRITE
                LCALL   TRANSMIT
                INC     DPTR
                DJNZ    R0,U_DISP2

                MOV     R0,#28H
                MOV     A,@R0            ; load number of characters
                MOV     R6,A             ; in 28H in reg R6
                DEC     R6
                MOV     R0,#81H          ; get start address
                MOV     A,@R0
                LCALL   HEXTODEC
                LCALL   TRANS_3_CH

                MOV     A,#'v'
                LCALL   WRITE
                LCALL   TRANSMIT
                MOV     A,#0DH
                LCALL   TRANSMIT
                CLR     TI
                MOV     R0,B            ; restore old reg R0 content

OPERATE:        MOV     R1,#15H         ; 15H - 18H
                CJNE    R2,#04H,THREE
                DEC     R2
                DEC     R0
                DEC     R0
                DEC     R0
                MOV     A,#0AH
BBB:            DEC     R2
                MOV     B,#04H
                DIV     AB
                MOV     @R1,A
                INC     R1
                MOV     A,#0AH
                MUL     AB
                INC     R0
                CJNE    R2,#00H,XXX
                JMP     OK
XXX:            ADD     A,@R0
                JMP     BBB
STOP_U2:        JMP     STOP_U
THREE:          CJNE    R2,#03H,TWO
```

```
            DEC     R0
WWW:        DEC     R0
CCC:        DEC     R0
            MOV     A,@R0
            JMP     BBB
TWO:        CJNE    R2,#02H,ONE
            INC     R2
            JMP     WWW
ONE:        CJNE    R2,#01H,OK
            INC     R2
            INC     R2
            JMP     CCC
OK:         LCALL   CHANGE
            MOV     P0,A
            CLR     P3.6
            LCALL   DELAY_1MS
            SETB    P3.6
            SETB    P2.4
            SETB    P2.5
            SETB    P2.6
            SETB    P2.7
            MOV     A,29H
            CJNE    A,#00H,UB1
            MOV     2FH,28H
STOP_U:     RET
UB1:        CJNE    A,#01H,UB2
            MOV     2EH,28H
            RET
UB2:        CJNE    A,#02H,UB3
            MOV     2DH,28H
            RET
```

```
UB3:        CJNE    A,#03H,UB4
            MOV     2CH,28H
UB4:        RET

CHANGE:     MOV     R1,#15H
            MOV     B,#64H
            MOV     A,@R1
            MUL     AB
            MOV     2BH,A
            INC     R1
            MOV     B,#0AH
            MOV     A,@R1
            MUL     AB
            INC     R1
            ADD     A,2BH
            ADD     A,@R1
            RET
BLOCK:      CJNE    A,#00H,BLOCK1
            MOV     R1,#56H
            CLR     P2.4
            RET
BLOCK1:     CJNE    A,#01H,BLOCK2
            MOV     R1,#5CH
            CLR     P2.5
            RET
BLOCK2:     CJNE    A,#02H,BLOCK3
            MOV     R1,#62H
            CLR     P2.6
            RET
BLOCK3:     CJNE    A,#03H,NOBLOCK
            MOV     R1,#68H
```

```
                CLR     P2.7
NOBLOCK:        RET


;************************************************
; COMMANDV  :  READ ANALOG INPUT
;  DATA ADDRESS USE : 90H  ANALOG INPUT VALUE
;               A0H  MAXIMUM INPUT VALUE
;               B0H  ANALOG INPUT LIMIT
;************************************************
COMMANDV:       LCALL   CLR_SCR
                SETB  TI

                MOV     R4,#00
                MOV     R3,#00010000B
                MOV     R7,#8
V_LOOP:         MOV     R0,#D0H

V_LOOP2:        MOV     A,R3            ;PC1,PC2,PC3 = 0  PC4 = 1
                MOV     DPTR,#AIN_C
                MOVX    @DPTR,A          ; OUT ADDRESS

                CLR     ACC.4
                MOVX    @DPTR,A
                LCALL   DELAY_1MS        ; ¯_
                SETB    ACC.4
                MOVX    @DPTR,A          ; ¯_-
                LCALL   DELAY_1MS

                CLR     ACC.4
                MOVX    @DPTR,A
```

```
            LCALL   DELAY_1MS        ; —_-_

            MOV     DPTR,#AIN_A
            MOVX    A,@DPTR         ; READ DATA
            MOV     @R0,A
            INC     R0
            DJNZ    R7,V_LOOP2
            MOV     A,#0DH
            LCALL   TRANSMIT

GET_TYPE:   LCALL   RECV
            CJNE    A,#'T',GET_TYPE
            MOV     A,#0DH
            LCALL   TRANSMIT
            LCALL   RECV
            MOV     R0,#E0H
            SUBB    A,#30H
            MOV     @R0,A
            MOV     A,#0DH
            LCALL   TRANSMIT

            MOV     A,#90H
            ADD     A,R4
            LCALL   AVERAGE
            MOV     R0,A
            MOV     @R0,A
            MOV     R1,#D0H
            MOV     A,@R1
            MOV     @R0,A
            MOV     A,#0DH
            LCALL   TRANSMIT
```

```
W_CHAR:         LCALL   RECV
                CJNE    A,#'W',W_CHAR
                LCALL   LIMIT

                JMP     L_CHAR

V_CONTI:        JMP     V_LOOP

L_CHAR:         LCALL   RECV
                CJNE    A,#'L',L_CHAR           ; 'L' character from PC to request
                MOV     A,#0DH                  ; the limit value
                LCALL   TRANSMIT
                MOV     A,#B0H
                ADD     A,R4
                MOV     R0,A
                LCALL   CNVT_TBL                ; convert and transmit limit value
                LCALL   TRANS_X_CH

V_CHAR:         LCALL   RECV
                CJNE    A,#'V',V_CHAR           ; 'V' character from PC to request
                MOV     A,#0DH                  ; the current value
                LCALL   TRANSMIT

                MOV     A,#90H                  ; get and convert data
                ADD     A,R4
                MOV     R0,A                    ; get each data store address
                LCALL   CNVT_TBL
                LCALL   TRANS_X_CH
```

```
M_CHAR:         LCALL   RECV
                CJNE    A,#'M',M_CHAR       ; 'M' character from PC to request
                MOV     A,#0DH             ; the maximum value
                LCALL   TRANSMIT
                LCALL   CAL_MAX             ; calculate maximum value
                LCALL   CNVT_TBL
                LCALL   TRANS_X_CH

                MOV     A,R3
                SETB    ACC.4
                INC     A
                INC     A              ; change to the next analog input card
                MOV     R3,A
                INC     R4               ; until equal to AIN_NO (in address 70H)
                MOV     A,R4
                CJNE    A,70H,V_CONTI
V_END:          CLR     TI
                RET

AVERAGE:        PUSH    ACC
                PUSH    DPL
                PUSH    DPH
                MOV     DPTR,#TEMP
                MOV     R1,#7
                CLR     CY
                MOV     R0,#D0H
                MOV     A,@R0              ; get first value from address D0h
                MOV     DPL,A              ; store in DPL
                INC     R0
AVG_LOOP:       MOV     A,@R0              ; get second value
                ADD     A,DPL              ; add first and second value
```

```
            MOVX    @DPTR,A          ; CY flay set if answer over 8 bit
            CLR     A
            ADDC    A,DPH
            MOVX    @DPTR,A          ; store 16 bit answer in DPTR
            INC     R0             ; get next value until
            DJNZ    R1,AVG_LOOP      ; end of value
            MOVX    A,@DPTR
            RRC     A
            RRC     A
            RRC     A              ; divide DPH by 8 the average store
            MOV     A,DPL            ; in DPL
            MOV     R0,#D0H
            MOV     @R0,A            ; store average value in address D0h
            POP     DPH
            POP     DPL
            POP     ACC
            RET

LIMIT:      MOV     A,#B0H         ; get limit from address B0H
            ADD     A,R4
            MOV     R0,A

            MOV     A,#90H         ; get current value
            ADD     A,R4
            MOV     R1,A
            MOV     A,@R0          ; limit value in ACC
            CLR     CY
            SUBB    A,@R1          ; compare current value to limit
            JC      OVER_LIMIT       ; jump if current value over limit
            JMP     LIMIT_Y
```

```
OVER_LIMIT:     MOV     A,#'N'
                LCALL   TRANSMIT
                LCALL   CLR_SCR
                MOV     DPTR,#WARNING1
                MOV     R1,#34
                MOV     R2,#00
                MOV     A,#00
                LCALL   GOTOXY
CHK_LINEW:      CJNE    R2,#11,W_LINE2
                MOV     A,#40H
                LCALL   GOTOXY
                JMP     WARN
W_LINE2:        CJNE    R2,#25,WARN
                MOV     A,#10H
                LCALL   GOTOXY
WARN:           CLR     A
                MOVC    A,@A+DPTR
                LCALL   WRITE
                INC     DPTR
                INC     R2
                DJNZ    R1,CHK_LINEW
                JMP     END_LIMIT

LIMIT_Y:        MOV     A,#'Y'
                LCALL   TRANSMIT
END_LIMIT:      MOV     A,#0DH
                LCALL   TRANSMIT
                RET

CAL_MAX:        MOV     A,#A0H          ; start address to store maximum value
                ADD     A,R4
                MOV     R0,A
                MOV     A,#90H          ; get new value
                ADD     A,R4
                MOV     R1,A
                MOV     A,@R0           ; load old MAX value to ACC
                CLR     CY              ; clear carry flag
                SUBB    A,@R1           ; compare old MAX value with new value
                JC      SWAP            ; jump if old MAX value < new value
                JMP     END_MAX
SWAP:           MOV     A,@R1
                MOV     @R0,A
END_MAX:        RET

;*********************************************
; Convert digital value to analog value
; Input  :  DPTR = start table address
; Output :  DPTR = point to 1st character
; Register : ACC,B,R0
;*********************************************
CNVT_TBL:       PUSH    ACC
                MOV     R1,#E0H                 ; get analog type from addr. E0H
                MOV     A,@R1
TMP:            CJNE    A,#0,RES
                MOV     DPTR,#TMP_TBL
                JMP     CNVT
RES:            CJNE    A,#1,HUM
                MOV     DPTR,#RES_TBL
                JMP     CNVT
HUM:            CJNE    A,#2,VOLT
                MOV     DPTR,#HUM_TBL
                JMP     CNVT
```

```
VOLT:           CJNE    A,#3,SPED
                MOV     DPTR,#VOLT_TBL
                JMP     CNVT
SPED:           MOV     DPTR,#SPED_TBL
CNVT:           MOV     A,@R0
                CLR     CY
                MOV     B,#6            ; load no. of ascii char to B
                MUL     AB
                ADD     A,DPL
                MOV     DPL,A
                MOV     A,B
                ADDC    A,DPH
                MOV     DPH,A
END_CNVT:       POP     ACC
                RET

TRANS_X_CH:     PUSH    ACC             ; transmit character to PC
                MOV     R0,#6           ; which number of char sotore in R0
TRANS_CH:       CLR     A
                MOVC    A,@A+DPTR
                LCALL   TRANSMIT
                INC     DPTR
                DJNZ    R0,TRANS_CH
                MOV     A,#0DH
                LCALL   TRANSMIT          ; sent CR to synchronous the PC
                POP     ACC
                RET
```

```
;*************************************
;           BLOCKINPUT
;       READ CARD NO. FROM KEYBOARD
;       CARD VALUE (0-3) IN 80H
;       REGISTER USED A,R0
;*************************************
CARDIP:         MOV     R5,#04H
                JNB     P3.3,$          ; wait until first key pressed
                JB      P3.3,$
                LCALL   CHK_KEY         ; card input no.in ACC
                MOV     R0,#80H
                MOV     @R0,A           ; store card input in 80H
                RET


;*********************************************
;   RECIVER COMMAND IN 40,41,42,43,44
;   ADDRESS LOW BIT PARAMETER1 IN REG R1
;   ADDRESS LOW BIT PARAMETER2 IN REG R0
;           USE REG A,R1,R0
;*********************************************
MRECV:          LCALL   DELAY_1MS
                MOV     R0,#3FH
DDD:            LCALL   RECV
                INC     R0
                CJNE            A,#2CH,COMEND
                MOV     @R0,A
                DEC     R0
                MOV     B,R0
                MOV     R1,B
                INC     R0
                JMP     DDD
YYY:            SUBB    A,#30H
                MOV     @R0,A
                JMP     DDD
COMEND:         CJNE    A,#0DH,YYY
                DEC     R0
END_MRECV:      RET

RECV:           CLR     RI
                JNB     RI,$
                CLR     RI
                MOV     A,SBUF
                RET

TRANSMIT:       LCALL   DELAY_1MS
                CLR     TI
                MOV     SBUF,A
                JNB     TI,$
                CLR     TI
                RET

;*********************************************
;           SUBRUTEEN PARA1
;     REN DATA DECMAL TO HEXDECMAL
;     DATA REN HEXDECMAL INTO REG A
;          USE REG A,R1,R7
;*********************************************
PARA1:          MOV     A,R1
                MOV     R1,#3FH
                MOV     @R1,#2CH
                MOV     R1,A
                MOV     A,@R1
                MOV     R7,A
                DEC     R1
                MOV     A,@R1
                CJNE    A,#2CH,EXIT
                MOV     A,#00H
EXIT:           MOV     B,#0AH
                MUL     AB
RENDATA:        ADD     A,R7
                RET


;*********************************************
;           SUBRUTEEN PARA2
;      REN DATA DECMAL TO HEXDECMAL
;      DATA REN HEXDECMAL INTO REG A
;           USE REG A,R0,R6,R7
;*********************************************
PARA2:          MOV     A,@R0
                MOV     R7,A
                DEC     R0
                MOV     A,@R0
                CJNE    A,#2CH,AEXIT
                MOV     R6,#00H
                MOV     A,#00H
                JMP     CEXIT
AEXIT:          MOV     B,#0AH
                MUL     AB
                MOV     R6,A
                DEC     R0
                MOV     A,@R0
                CJNE    A,#2CH,BEXIT
                MOV     A,#00H
                JMP     CEXIT
BEXIT:          MOV     B,#64H
                MUL     AB
CEXIT:          ADD     A,R7
                ADD     A,R6
                RET


;*********************************************
;            SUBRUTEEN HEXTODEC
;       CHANGE HEXDECIMAL TO DECIMAL
;       PARAMETER1 <00 TO FF> IN REGISTER R0
; ANSWER <0 TO 255> IN 51H,52H,53H,54H
;            USE REG A,B,R0,R1
;*********************************************
HEXTODEC:       MOV     A,@R0           ; get data from address
                MOV     B,#64H
                DIV     AB              ; divide ACC by 100
                MOV     R1,A            ; get first data in R1
                MOV     A,B             ; get data in reg B to ACC
                MOV     B,#0AH
                DIV     AB              ; divide ACC by 10
                ADD     A,#30H
                XCH     A,R1
                ADD     A,#30H
                XCH     A,R1
                XCH     A,B
                ADD     A,#30H
                XCH     A,B
```

```
            MOV     R0,#54H
            MOV     @R0,#0DH       ; save CR character to 54H
            DEC     R0
            MOV     @R0,B          ; save third data to 53H
            DEC     R0
            MOV     @R0,A          ; save second data to 52H
            DEC     R0
            MOV     A,R1
            MOV     @R0,A          ; save first data to 51H
            RET

WRITE_3_CH: MOV     A,51H
            LCALL   WRITE
            LCALL   TRANSMIT
            MOV     A,52H
            LCALL   WRITE
            LCALL   TRANSMIT
            MOV     A,53H
            LCALL   WRITE
            LCALL   TRANSMIT
            MOV     A,#0DH
            LCALL   TRANSMIT
            RET

LCD_3_CH:   PUSH    ACC
            MOV     A,51H
            LCALL   WRITE
            MOV     A,52H
            LCALL   WRITE
            MOV     A,53H
            LCALL   WRITE
```

```
                POP     ACC
                RET

TRANS_3_CH:     MOV     A,51H
                LCALL   TRANSMIT
                MOV     A,52H
                LCALL   TRANSMIT
                MOV     A,53H
                LCALL   TRANSMIT
                MOV     A,#0DH
                LCALL   TRANSMIT
                RET

RTC_INIT:       MOV     DPTR,#CREG_F
                MOV     A,#01H
                MOVX    @DPTR,A
                MOV     A,#04H          ; set 24 hour mode
                MOVX    @DPTR,A
                MOV     A,#04H
                MOV     DPTR,#CREG_E
                MOVX    @DPTR,A
                MOV     A,#00H
                MOV     DPTR,#CREG_D
                MOVX    @DPTR,A

                MOV     A,#SET_YEAR
                MOV     DPTR,#YEAR1
                LCALL   SET_TIME

                MOV     A,#SET_MONTH
                MOV     DPTR,#MONTH1
```

```
            LCALL   SET_TIME

            MOV     A,#SET_DAY
            MOV     DPTR,#DAY1
            LCALL   SET_TIME

            MOV     A,#SET_HOUR
            MOV     DPTR,#HOUR1
            LCALL   SET_TIME

            MOV     A,#SET_MINUTE
            MOV     DPTR,#MIN1
            LCALL   SET_TIME

            MOV     A,#SET_SEC
            MOV     DPTR,#SEC1
            LCALL   SET_TIME
            RET

SET_TIME:   MOVX    @DPTR,A
            LCALL   SHIFT
            INC     DPTR
            MOVX    @DPTR,A
            RET

TIME_DSP:   LCALL   RTC_TITLE
            MOV     A,#05H
            LCALL   GOTOXY
            MOV     DPTR,#BUFFER+3
            LCALL   RTC_DISP
```

```
MOV      A,#07H
LCALL    GOTOXY
MOV      A,#'/'
LCALL    WRITE
MOV      DPTR,#BUFFER+4
LCALL    RTC_DISP

MOV      A,#0AH
LCALL    GOTOXY
MOV      A,#'/'
LCALL    WRITE
MOV      DPTR,#BUFFER+5
LCALL    RTC_DISP

MOV      A,#15H
LCALL    GOTOXY
MOV      DPTR,#BUFFER+2
LCALL    RTC_DISP
MOVX     A,@DPTR

MOV      A,#17H
LCALL    GOTOXY
MOV      A,#':'
LCALL    WRITE
MOV      DPTR,#BUFFER+1
LCALL    RTC_DISP

MOV      A,#1AH
LCALL    GOTOXY
MOV      A,#':'
LCALL    WRITE
```

```
                MOV     DPTR,#BUFFER
                LCALL   RTC_DISP
                RET

RTC_DISP:       MOVX    A,@DPTR
                MOV     R0,#90H
                MOV     @R0,A
                LCALL   HEXTODEC
                MOV     A,52H
                LCALL   WRITE
                MOV     A,53H
                LCALL   WRITE
                RET


RTC_TITLE:      MOV     A,#00H
                LCALL   GOTOXY
                MOV     R1,#04H
                MOV     DPTR,#DATE
DATE1:          CLR     A
                MOVC    A,@A+DPTR
                LCALL   WRITE
                INC     DPTR
                DJNZ    R1,DATE1
                MOV     A,#10H
                LCALL   GOTOXY
                MOV     R1,#4H
                MOV     DPTR,#TIME
TIME1:          CLR     A
                MOVC    A,@A+DPTR
                LCALL   WRITE
                INC     DPTR
                DJNZ    R1,TIME1
                RET


DELAY_1MS:      MOV     60H,#1
                MOV     61H,#0
DELAY1:         MOV     62H,#226
MIL1:           NOP
                NOP
                DJNZ    62H,MIL1
                NOP
                NOP
                DJNZ    60H,DELAY1
                PUSH    ACC
                MOV     A,61H
                CJNE    A,#0,HIDOWN
                POP     ACC
                SJMP    DONE
HIDOWN:         DEC     A
                MOV     61H,A
                POP     ACC
                SJMP    DELAY1
DONE:           RET


CLR_SCR:        PUSH    DPL
                PUSH    DPH
                PUSH    ACC
                MOV     DPTR,#COMMAND
                MOV     A,#1
                MOVX    @DPTR,A
                LCALL   WAITBF
                POP     ACC
                POP     DPH
                POP     DPL
                RET

PORT_INIT:      MOV     A,#90H
                MOV     DPTR,#CTRL
                MOVX    @DPTR,A             ; define 8255 for analog input card
                MOV     R0,#72H
                MOV     A,@R0
                JZ      ALL_OUTPUT
                DEC     A
                MOV     B,#3
                DIV     AB
                JMP     A_IS_0

ALL_OUTPUT:     MOV     DPTR,#CTRL_1        ; this loop is done if digital
                MOV     A,#80H              ; input no. is 0
                MOVX    @DPTR,A             ; ( digital output is 12 )
                MOV     DPTR,#CTRL_2
                MOVX    @DPTR,A
                MOV     DPTR,#CTRL_3
                MOVX    @DPTR,A
                MOV     DPTR,#CTRL_4
                MOVX    @DPTR,A
                RET

A_IS_0:         CJNE    A,#00,A_IS_1        ; this loop is done if digital
                MOV     A,#82H              ; input number is between 1 - 3
                ADD     A,B             ; ( digital output is  9 - 11 )
                MOV     R0,A
                MOV     DPTR,#CTRL_1
                MOV     A,@R0
                MOVX    @DPTR,A
                MOV     R3,#00
                INC     DPTR
                INC     DPTR
                INC     DPTR
                INC     DPTR
                MOV     A,#80H
LOOP_0_3:       MOVX    @DPTR,A
                INC     R3
                INC     DPTR
                INC     DPTR
                INC     DPTR
                INC     DPTR
                CJNE    R3,#3,LOOP_0_3
                MOV     DPTR,#CTRL_4
                RET

A_IS_1:         CJNE    A,#01,A_IS_2
                MOV     A,#9BH
                MOV     DPTR,#CTRL_1
                MOVX    @DPTR,A
                MOV     A,#82H
                ADD     A,B
                MOV     R0,A
                MOV     DPTR,#CTRL_2
                MOV     A,@R0
                MOVX    @DPTR,A
                MOV     DPTR,#CTRL_3
                MOV     A,#80H
                MOVX    @DPTR,A
                MOV     DPTR,#CTRL_4
                MOVX    @DPTR,A
                RET

A_IS_2:         CJNE    A,#02,A_IS_3
                MOV     A,#9BH
                MOV     DPTR,#CTRL_1
                MOVX    @DPTR,A
                MOV     DPTR,#CTRL_2
                MOVX    @DPTR,A
                MOV     A,#82H
                ADD     A,B
                MOV     R0,A
                MOV     DPTR,#CTRL_3
                MOV     A,@R0
                MOVX    @DPTR,A
                MOV     A,#80H
                MOV     DPTR,#CTRL_4
                MOVX    @DPTR,A
                RET

A_IS_3:         MOV     R3,#00
                MOV     DPTR,#CTRL_1
                MOV     A,#9BH
LOOP_1_3:       MOVX    @DPTR,A
                INC     R3
                INC     DPTR
                INC     DPTR
                INC     DPTR
                INC     DPTR
                CJNE    R3,#3,LOOP_1_3
                MOV     A,#82H
                ADD     A,B
                MOV     R0,A
                MOV     A,@R0
                MOVX    @DPTR,A
                RET

TITLE:          MOV     A,#40H
                LCALL   GOTOXY
                MOV     R0,#16
                MOV     DPTR,#TABLE1
TITL1:          CLR     A
                MOVC    A,@A+DPTR
                LCALL   WRITE
                INC     DPTR
                DJNZ    R0,TITL1

                MOV     A,#10H
                LCALL   GOTOXY
                MOV     DPTR,#TABLE2
                MOV     R0,#13
TIT2:           CLR     A
                MOVC    A,@A+DPTR
                LCALL   WRITE
                INC     DPTR
                DJNZ    R0,TIT2
                RET
```

```
; *** Write ASCII to LCD ***

WRITE:          PUSH    DPL
                PUSH    DPH
                MOV     DPTR,#WRITEDATA
                MOVX    @DPTR,A
                LCALL   WAITBF
                POP     DPH
                POP     DPL
                RET
; *** Wait for ready
; *** by mean of check busy flag ***
WAITBF:         PUSH    DPL
                PUSH    DPH
                PUSH    ACC
                MOV     DPTR,#READBUSY
RDY1:           MOVX    A,@DPTR
                JB      ACC.7,RDY1              ; Busy Flag
                POP     ACC
                POP     DPH
                POP     DPL
                RET

LCD_INIT:       PUSH    DPL
                PUSH    DPH
                MOV     DPTR,#COMMAND
                MOV     A,#38H                  ; 8 bit, 2 line, 5x7 dot
                MOVX    @DPTR,A
                LCALL   WAITBF
                MOV     A,#0CH
                MOVX    @DPTR,A
                LCALL   WAITBF
                MOV     A,#6                ; increment cursor
                MOVX    @DPTR,A
                LCALL   WAITBF
                MOV     A,#1                ; clear and home
                MOVX    @DPTR,A
                LCALL   WAITBF
                POP     DPH
                POP     DPL
                RET

GOTOXY:         PUSH    DPL
                PUSH    DPH
                PUSH    ACC
                MOV     DPTR,#COMMAND
                SETB    ACC.7               ; set DD ram instruction
                MOVX    @DPTR,A
                LCALL   WAITBF
                POP     ACC
                POP     DPH
                POP     DPL
                RET

LINEI1:         DB      ' INITIALIZE '
LINEI2:         DB      'ANALOG I/P '
LINEI3:         DB      'ANALOG O/P '
LINEI4:         DB      'DIGITAL I/P '
LINEI5:         DB      'DIGITAL O/P '

LINEA1:         DB      ' WRITE LOGIC '
LINEA2:         DB      ' TO DIGITAL O/P '
```

```
LINEA3:         DB      'CARD No.'
LINEA4:         DB      ' BIT '

LINE_A1:        DB      'DIGITAL OUTPUT '
LINE_A2:        DB      'CARD No.'
LINE_A3:        DB      ' BIT '
LINE_A4:        DB      'VALUE IS : '
LINE_A5:        DB      'READ DIGITAL O/P ENTER BIT TO BEREAD IN DECIMAL '
                DB      'RANGE (0-31)'

LINEB1:         DB      'WRITE VALUE '
LINEB2:         DB      'TO DIGITAL O/P'
LINEB3:         DB      'CARD No.'

LINE_B1:        DB      ' DIGITAL OUTPUT'
LINE_B2:        DB      ' CARD No.'
LINE_B3:        DB      ' VALUE IS : '
LINE_B4:        DB      'READ DIGITAL O/P ENTER CARD TO BEREAD IN DECIMAL '
                DB      'RANGE (0-3)'

LINEC1:         DB      'WRITE DIGITAL OUTPUT VALUE 00000000 B'
LINEC2:         DB      'NO DIGITAL OUTPUT CARD INSTALLED'
LINED1:         DB      'WRITE DIGITAL OUTPUT VALUE 11111111 B'

LINEE1:         DB      'DIGITAL INPUT '
LINEE2:         DB      'CARD No.'
LINEE3:         DB      'BIT '
LINEE4:         DB      'VALUE IS : '
LINEE5:         DB      'READ DIGITAL I/P ENTER BIT TO BEREAD IN DECIMAL '
                DB      'RANGE (0-31)'
```

```
LINEF1:      DB    ' DIGITAL INPUT'
LINEF2:      DB    ' CARD No.'
LINEF3:      DB    ' VALUE IS : '
LINEF4:      DB    'READ DIGITAL I/P ENTER CARD TO BEREAD IN DECIMAL '
             DB    'RANGE (0-3)'

LINER1:      DB    'RESET ALL SYSTEM'

LINEU1:      DB    'WRITE ANALOG O/P CARD No.'
LINEU2:      DB    'VALUE : '

LINE_U1:     DB    'ANALOG OUTPUT CARD NO.'
LINE_U2:     DB    'VALUE IS '

LINEV1:      DB    'ANALOG INPUT CHANNEL NO.'
LINEV2:      DB    'VALUE IS '

WARNING1:    DB    'WARNING....VALUE REACHED THE LIMIT'
WARNING2:    DB    'BIT/CARD ENTERED EXCEED THE CARD INSTALLED'
DATE:        DB    'DATE'
TIME:        DB    'TIME'
BUFFER:      DS    6

TABLE1:      DB    'DATA ACQUISITION'
TABLE2:      DB    '  Version 1.0'
```

```
TMP_TBL:        DB      '-273.2','-271.2','-269.2','-267.3','-265.3'
            DB      '-263.4','-261.4','-259.4','-257.5','-255.5'
            DB      '-253.6','-251.6','-249.6','-247.7','-245.7'
            DB      '-243.8','-241.8','-239.8','-237.9','-235.9'
            DB      '-234.0','-232.0','-230.0','-228.1','-226.1'
            DB      '-224.2','-222.2','-220.2','-218.3','-216.3'
            DB      '-214.4','-212.4','-210.4','-208.5','-206.5'
            DB      '-204.6','-202.6','-200.6','-198.7','-196.7'
            DB      '-194.8','-192.8','-190.8','-188.9','-186.9'
            DB      '-185.0','-183.0','-181.0','-179.1','-177.1'
            DB      '-175.2','-173.2','-171.2','-169.3','-167.3'
            DB      '-165.4','-163.4','-161.4','-159.5','-157.5'
            DB      '-155.6','-153.6','-151.6','-149.7','-147.7'
            DB      '-145.8','-143.8','-141.8','-139.9','-137.9'
            DB      '-136.0','-134.0','-132.0','-130.1','-128.1'
            DB      '-126.2','-124.2','-122.2','-120.3','-118.3'
            DB      '-116.4','-114.4','-112.4','-110.5','-108.5'
            DB      '-106.6','-104.6','-102.6','-100.7','- 98.7'
            DB      '- 96.8','- 94.8','- 92.8','- 90.9','- 88.9'
            DB      '- 87.0','- 85.0','- 83.0','- 81.1','- 79.1'
            DB      '- 77.2','- 75.2','- 73.2','- 71.3','- 69.3'
            DB      '- 67.4','- 65.4','- 63.4','- 61.5','- 59.5'
            DB      '- 57.6','- 55.6','- 53.6','- 51.7','- 49.7'
            DB      '- 47.8','- 45.8','- 43.8','- 41.9','- 39.9'
            DB      '- 38.0','- 36.0','- 34.0','- 32.1','- 30.1'
            DB      '- 28.2','- 26.2','- 24.2','- 22.3','- 20.3'
            DB      '- 18.4','- 16.4','- 14.4','- 12.5','- 10.5'
            DB      '-  8.6','-  6.6','-  4.6','-  2.7','-  0.7'
            DB      '+  1.2','+  3.1','+  5.1','+  7.0','+  9.0'
            DB      '+ 11.0','+ 12.9','+ 14.9','+ 16.8','+ 18.8'
            DB      '+ 20.8','+ 22.7','+ 24.7','+ 26.6','+ 28.6'
            DB      '+ 30.6','+ 32.5','+ 34.5','+ 36.4','+ 38.4'
            DB      '+ 40.4','+ 42.3','+ 44.3','+ 46.2','+ 48.2'
            DB      '+ 50.2','+ 52.1','+ 54.1','+ 56.0','+ 58.0'
            DB      '+ 60.0','+ 61.9','+ 63.9','+ 65.8','+ 67.8'
            DB      '+ 69.8','+ 71.7','+ 73.7','+ 75.6','+ 77.6'
            DB      '+ 79.6','+ 81.5','+ 83.5','+ 85.4','+ 87.4'
            DB      '+ 89.4','+ 91.3','+ 93.3','+ 95.2','+ 97.2'
            DB      '+ 99.2','+101.1','+103.1','+105.0','+107.0'
            DB      '+109.0','+110.9','+112.9','+114.8','+116.8'
            DB      '+118.8','+120.7','+122.7','+124.6','+126.6'
            DB      '+128.6','+130.5','+132.5','+134.4','+136.4'
            DB      '+138.4','+140.3','+142.3','+144.2','+146.2'
            DB      '+148.2','+151.1','+152.1','+154.0','+156.0'
            DB      '+158.0','+159.9','+161.9','+163.8','+165.8'
            DB      '+167.8','+169.7','+171.7','+173.6','+175.6'
            DB      '+177.6','+179.5','+181.5','+183.4','+185.4'
            DB      '+187.6','+189.3','+191.3','+193.2','+195.2'
            DB      '+197.2','+199.1','+201.1','+203.0','+205.0'
            DB      '+207.0','+208.9','+210.9','+212.8','+214.8'
            DB      '+216.8','+218.7','+220.7','+222.6','+224.6'
            DB      '+226.6'

VOLT_TBL:   DB      '000000','0.0196','0.0392','0.0588','0.0784'
            DB      '0.0980','0.1176','0.1373','0.1569','0.1765'
            DB      '0.1961','0.2157','0.2353','0.2549','0.2745'
            DB      '0.2941','0.3137','0.3333','0.3529','0.3725'
            DB      '0.3922','0.4118','0.4314','0.4510','0.4706'
            DB      '0.4901','0.5098','0.5294','0.5490','0.5686'
            DB      '0.5882','0.6078','0.6275','0.6471','0.6666'
            DB      '0.6863','0.7059','0.7255','0.7451','0.7647'
            DB      '0.7843','0.8039','0.8235','0.8431','0.8627'
DB      '0.8824','0.9020','0.9216','0.9412','0.9608'
DB      '0.9804','1.0000','1.0196','1.0392','1.0588'
DB      '1.0784','1.0980','1.1176','1.1373','1.1569'
DB      '1.1765','1.1961','1.2157','1.2353','1.2549'
DB      '1.2745','1.2941','1.3137','1.3333','1.3529'
DB      '1.3725','1.3922','1.4118','1.4314','1.4510'
DB      '1.4706','1.4902','1.5098','1.5294','1.5490'
DB      '1.5686','1.5824','1.6078','1.6275','1.6471'
DB      '1.6667','1.6863','1.7059','1.7255','1.7451'
DB      '1.7647','1.7843','1.8039','1.8235','1.8431'
DB      '1.8627','1.8824','1.9020','1.9216','1.9412'
DB      '1.9608','1.9804','2.0000','2.0196','2.0392'
DB      '2.0589','2.0784','2.0980','2.1176','2.1373'
DB      '2.1569','2.1765','2.1961','2.2157','2.2353'
DB      '2.2549','2.2745','2.2941','2.3137','2.3333'
DB      '2.3529','2.3725','2.3922','2.4118','2.4314'
DB      '2.4510','2.4706','2.4902','2.5098','2.5294'
DB      '2.5490','2.5686','2.5882','2.6078','2.6275'
DB      '2.6471','2.6667','2.6863','2.7059','2.7255'
DB      '2.7451','2.7647','2.7843','2.8039','2.8235'
DB      '2.8431','2.8627','2.8824','2.9020','2.9216'
DB      '2.9412','2.9608','2.9804','3.0000','3.0196'
DB      '3.0392','3.0588','3.0784','3.0980','3.1176'
DB      '3.1373','3.1569','3.1765','3.1961','3.2157'
DB      '3.2353','3.2550','3.2745','3.2941','3.3137'
DB      '3.3333','3.3530','3.3725','3.3922','3.4112'
DB      '3.4314','3.4510','3.4706','3.4902','3.5098'
DB      '3.5294','3.5490','3.5686','3.5882','3.6078'
DB      '3.6275','3.6471','3.6667','3.6863','3.7059'
DB      '3.7255','3.7451','3.7647','3.7843','3.8039'
DB      '3.8235','3.8431','3.8627','3.8824','3.9020'
```

```
          DB   '3.9216','3.9412','3.9608','3.9804','4.0000'
          DB   '4.0196','4.0392','4.0589','4.0784','4.0980'
          DB   '4.1176','4.1373','4.1569','4.1765','4.1961'
          DB   '4.2157','4.2353','4.2549','4.2745','4.2941'
          DB   '4.3137','4.3333','4.3529','4.3725','4.3922'
          DB   '4.4118','4.4314','4.4510','4.4706','4.4902'
          DB   '4.5098','4.5294','4.5490','4.5686','4.5882'
          DB   '4.6078','4.6275','4.6471','4.6667','4.6863'
          DB   '4.7059','4.7255','4.7451','4.7647','4.7843'
          DB   '4.8039','4.8235','4.8431','4.8627','4.8824'
          DB   '4.9020','4.9216','4.9412','4.9608','4.9804'
          DB   '5.0000'

RES_TBL:    DB   '000000','000001','000002','000003','000004'
          DB   '000005','000006','000007','000008','000009'
          DB   '000010','000011','000012','000013','000014'
          DB   '000015','000016','000017','000018','000019'
          DB   '000020','000021','000022','000023','000024'
          DB   '000025','000026','000027','000028','000029'
          DB   '000030','000031','000032','000033','000034'
          DB   '000035','000036','000037','000038','000039'
          DB   '000040','000041','000042','000043','000044'
          DB   '000045','000046','000047','000048','000049'
          DB   '000050','000051','000052','000053','000054'
          DB   '000055','000056','000057','000058','000059'
          DB   '000060','000061','000062','000063','000064'
          DB   '000065','000066','000067','000068','000069'
          DB   '000070','000071','000072','000073','000074'
          DB   '000075','000076','000077','000078','000079'
          DB   '000080','000081','000082','000083','000084'
          DB   '000085','000086','000087','000088','000089'
DB      '000090','000091','000092','000093','000094'
DB      '000095','000096','000097','000098','000099'
DB      '000100','000101','000102','000103','000104'
DB      '000105','000106','000107','000108','000109'
DB      '000110','000111','000112','000113','000114'
DB      '000115','000116','000117','000118','000119'
DB      '000120','000121','000122','000123','000124'
DB      '000125','000126','000127','000128','000129'
DB      '000130','000131','000132','000133','000134'
DB      '000135','000136','000137','000138','000139'
DB      '000140','000141','000142','000143','000144'
DB      '000145','000146','000147','000148','000149'
DB      '000150','000151','000152','000153','000154'
DB      '000155','000156','000157','000158','000159'
DB      '000160','000161','000162','000163','000164'
DB      '000165','000166','000167','000168','000169'
DB      '000170','000171','000172','000173','000174'
DB      '000175','000176','000177','000178','000179'
DB      '000180','000181','000182','000183','000184'
DB      '000185','000186','000187','000188','000189'
DB      '000190','000191','000192','000193','000194'
DB      '000195','000196','000197','000198','000199'
DB      '000200','000201','000202','000203','000204'
DB      '000205','000206','000207','000208','000209'
DB      '000210','000211','000212','000213','000214'
DB      '000215','000216','000217','000218','000219'
DB      '000220','000221','000222','000223','000224'
DB      '000225','000226','000227','000228','000229'
DB      '000230','000231','000232','000233','000234'
DB      '000235','000236','000237','000238','000239'
DB      '000240','000241','000242','000243','000244'
```

```
          DB      '000245','000246','000247','000248','000249'
          DB      '000250','000251','000252','000253','000254'
          DB      '000255'

HUM_TBL:     DB      '000000','000001','000002','000003','000004'
          DB      '000005','000006','000007','000008','000009'
          DB      '000010','000011','000012','000013','000014'
          DB      '000015','000016','000017','000018','000019'
          DB      '000020','000021','000022','000023','000024'
          DB      '000025','000026','000027','000028','000029'
          DB      '000030','000031','000032','000033','000034'
          DB      '000035','000036','000037','000038','000039'
          DB      '000040','000041','000042','000043','000044'
          DB      '000045','000046','000047','000048','000049'
          DB      '000050','000051','000052','000053','000054'
          DB      '000055','000056','000057','000058','000059'
          DB      '000060','000061','000062','000063','000064'
          DB      '000065','000066','000067','000068','000069'
          DB      '000070','000071','000072','000073','000074'
          DB      '000075','000076','000077','000078','000079'
          DB      '000080','000081','000082','000083','000084'
          DB      '000085','000086','000087','000088','000089'
          DB      '000090','000091','000092','000093','000094'
          DB      '000095','000096','000097','000098','000099'
          DB      '000100','000101','000102','000103','000104'
          DB      '000105','000106','000107','000108','000109'
          DB      '000110','000111','000112','000113','000114'
          DB      '000115','000116','000117','000118','000119'
          DB      '000120','000121','000122','000123','000124'
          DB      '000125','000126','000127','000128','000129'
          DB      '000130','000131','000132','000133','000134'
          DB      '000135','000136','000137','000138','000139'
          DB      '000140','000141','000142','000143','000144'
          DB      '000145','000146','000147','000148','000149'
          DB      '000150','000151','000152','000153','000154'
          DB      '000155','000156','000157','000158','000159'
          DB      '000160','000161','000162','000163','000164'
          DB      '000165','000166','000167','000168','000169'
          DB      '000170','000171','000172','000173','000174'
          DB      '000175','000176','000177','000178','000179'
          DB      '000180','000181','000182','000183','000184'
          DB      '000185','000186','000187','000188','000189'
          DB      '000190','000191','000192','000193','000194'
          DB      '000195','000196','000197','000198','000199'
          DB      '000200','000201','000202','000203','000204'
          DB      '000205','000206','000207','000208','000209'
          DB      '000210','000211','000212','000213','000214'
          DB      '000215','000216','000217','000218','000219'
          DB      '000220','000221','000222','000223','000224'
          DB      '000225','000226','000227','000228','000229'
          DB      '000230','000231','000232','000233','000234'
          DB      '000235','000236','000237','000238','000239'
          DB      '000240','000241','000242','000243','000244'
          DB      '000245','000246','000247','000248','000249'
          DB      '000250','000251','000252','000253','000254'
          DB      '000255'


SPED_TBL:   DB      '000000','000001','000002','000003','000004'
          DB      '000005','000006','000007','000008','000009'
          DB      '000010','000011','000012','000013','000014'
          DB      '000015','000016','000017','000018','000019'
DB      '000020','000021','000022','000023','000024'
DB      '000025','000026','000027','000028','000029'
DB      '000030','000031','000032','000033','000034'
DB      '000035','000036','000037','000038','000039'
DB      '000040','000041','000042','000043','000044'
DB      '000045','000046','000047','000048','000049'
DB      '000050','000051','000052','000053','000054'
DB      '000055','000056','000057','000058','000059'
DB      '000060','000061','000062','000063','000064'
DB      '000065','000066','000067','000068','000069'
DB      '000070','000071','000072','000073','000074'
DB      '000075','000076','000077','000078','000079'
DB      '000080','000081','000082','000083','000084'
DB      '000085','000086','000087','000088','000089'
DB      '000090','000091','000092','000093','000094'
DB      '000095','000096','000097','000098','000099'
DB      '000100','000101','000102','000103','000104'
DB      '000105','000106','000107','000108','000109'
DB      '000110','000111','000112','000113','000114'
DB      '000115','000116','000117','000118','000119'
DB      '000120','000121','000122','000123','000124'
DB      '000125','000126','000127','000128','000129'
DB      '000130','000131','000132','000133','000134'
DB      '000135','000136','000137','000138','000139'
DB      '000140','000141','000142','000143','000144'
DB      '000145','000146','000147','000148','000149'
DB      '000150','000151','000152','000153','000154'
DB      '000155','000156','000157','000158','000159'
DB      '000160','000161','000162','000163','000164'
DB      '000165','000166','000167','000168','000169'
DB      '000170','000171','000172','000173','000174'
```

```
          DB      '000175','000176','000177','000178','000179'
          DB      '000180','000181','000182','000183','000184'
          DB      '000185','000186','000187','000188','000189'
          DB      '000190','000191','000192','000193','000194'
          DB      '000195','000196','000197','000198','000199'
          DB      '000200','000201','000202','000203','000204'
          DB      '000205','000206','000207','000208','000209'
          DB      '000210','000211','000212','000213','000214'
          DB      '000215','000216','000217','000218','000219'
          DB      '000220','000221','000222','000223','000224'
          DB      '000225','000226','000227','000228','000229'
          DB      '000230','000231','000232','000233','000234'
          DB      '000235','000236','000237','000238','000239'
          DB      '000240','000241','000242','000243','000244'
          DB      '000245','000246','000247','000248','000249'
          DB      '000250','000251','000252','000253','000254'
          DB      '000255'

TEMP:        DS      2
          END
```

ภาคผนวก ข.

## โปรแกรมภาษา C ที่ใช้ควบคุมคอมพิวเตอร์

```
#include <conio.h>
#include <string.h>
#include <process.h>
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <config.h>

#define  PORT 0
#define  TRUE  1

#define  U_ARRO  72
#define  D_ARRO  80
#define  R_ARRO  77
#define  L_ARRO  75

#define  CR   13
#define  ESC  27
#define  INS  82
#define  F1   59
```

```
#define  MENU_WIDTH  39
#define  MAX_ITEM     5

#define  MAIN_WIDTH  15
#define  MAIN_MAX  6
#define  D_IO_NO   6
#define  KETOCEL 273.2

char FILEBUFF[] = {"THESIS.DAT"};
void  main_menu(),command();
void  display_menu(int left, int top, char *menu_item[],int width, int max, char *header,int x, int y);
void  setup_display(int left, int top, char *menu_item[],int width,int max, char *header,int x, int y);
void  draw_box(int left, int top, int width, int height);
char  get_code(void);
int   card1(char *st), sub2(char *st), menu_action();
int   ad_init();
void  act_line(int left, int top, char *menu_item[]);
void  del_line(int left, int top, char *menu_item[]);
void  cursor(), sport(),port_init(), wait();
void  nocursor(), init(), setup(), alarm();
void DrawHelp(char *help_buff);
void DrawSubHelp(int cur,int x,char *subhelp[]);
void ErrMsg(char *str);
void  d_port_scr(int port),  d_port_newscr(int port,int oport,int flag), d_port_oldscr(int port);
/* flag 0 CR clear value output no update oldscr, 1 ARR update oldscr*/
int   out_port(int port, int x, int y);
int   v_pos;

int   buffer0[80][25];         /*  save whole screen  */
int   buffer1[80][25];         /*  save main menu screen */
int   buffer2[80][25];         /*  save set up menu screen */
int   buffer3[80][25];
int   buffer4[80][25];
int   buffer5[80][25];
FILE  *fp;
int   A_D_VAL[24];            /* keep analog/digital card number and limit */
                                  /* A_D_VAL[0]  =  Analog input card        */
                                  /* A_D_VAL[1]  =  Analog output card        */
                                  /* A_D_VAL[2]  =  Digital input card       */
                                  /* A_D_VAL[3]  =  Digital output card       */
                                  /* A_D_VAL[4-15]  =  Analog input limit     */
char  *a_d_name[] = {"AIN_NO","AOUT_NO","DIN_NO","DOUT_NO",
                     "LIMIT#1","LIMIT#2","LIMIT#3","LIMIT#4",
                     "LIMIT#5","LIMIT#6","LIMIT#7","LIMIT#8",
                     "LIMIT#9","LIMIT#10","LIMIT#11","LIMIT#12"};


char *ana_type[] = {"ANALOG#1","ANALOG#2","ANALOG#3","ANALOG#4",
                    "ANALOG#5","ANALOG#6","ANALOG#7","ANALOG#8",
                    "ANALOG#9","ANALOG#10","ANALOG#11","ANALOG#12"};


char   *type_item[13] = {" Temperature ",
                         " Resistance  ",
                         " Humidity    ",
                         " Voltage     ",
                         " Speed       "};


char   main_help[] =
        {" \x18,\x19 and ENTER:  Select function.            "};


char   *sub_help1[] =
        {"  SET ANALOG/DIGITAL CARD.           ",
         "  RESET SYSTEM BUFFER.               ",
            " READ/WRITE DIGITAL INPUT OUTPUT. ",
            " READ ANALOG INPUT.              ",
            " SAVE TO SPECIFIC DRIVE/FILE.    ",
            " EXIT FROM SYSTEM.               "};

char   init_help[] =
         {" \x18,\x19 and F1: Select      ENTER: Change     ESC: Exit        "};
char   reset_help[] =
         {" \x18,\x19 and ENTER: Select                          ESC: Exit  "};

char   digit_help[] =
         {" \x1A,\x1B and F1: Select            ENTER: Change               ESC: Exit  "};

char   ana_help[] =
         {" \x18,\x19 and F1: Select Row  \x1A,\x1B and F1: Select Col.  ENTER: Change   ESC: Exit "};

int   A_TYPE[10];                            /* Analog sensor type */
char  temp[80];

int   main(void)
{
    int i,val;
    char  str[10];

    clrscr();
    port_init(PORT);
    gettext(1,1,80,24,buffer0);
    textbackground(1);textcolor(14);

    draw_box(1,1,80,24);
    nocursor();
```

```
if((fp = fopen("thesis.cfg","r+")) == NULL)
{
       fp = fopen("thesis.cfg","w+");
}
if ((fgets(temp,sizeof(temp),fp)) == NULL)
{
       setup();
}
else
 {
         fseek(fp,0,0);
         while (!feof(fp))
         {
             fscanf(fp,"%s %d",&str,&val);
             if (strcmp(str,"AIN_NO")==0)
                    A_D_VAL[0] = val;
             else if (strcmp(str,"AOUT_NO")==0)
                      A_D_VAL[1] = val;
                      else if (strcmp(str,"DIN_NO")==0)
                               A_D_VAL[2] = val;
                               else if (strcmp(str,"DOUT_NO")==0)
                                    A_D_VAL[3] = val;
                                    else {  for(i=0;i<A_D_VAL[0];i++)
                                                { if (strcmp(str,a_d_name[i+4]) == 0)
                                                          A_D_VAL[i+4] = val;
                                                }
                                                for(i=0;i<A_D_VAL[0];i++)
                                                { if (strcmp(str,ana_type[i]) == 0)
                                                      A_TYPE[i] = val;
                                                }
                                            }
```

```
                }
        }
        init();
        main_menu();
}

void    main_menu()
{

       char    *main_item[MAIN_WIDTH] =
                    {" INITIALIZE    ",
                    " I/O COMMAND   ",
                    " SHOW DIGITAL ",
                    " SHOW ANALOG   ",
                    " SAVE AS...    ",
                    " QUIT           "};

      int      main_width = MAIN_WIDTH;
      int      main_max = MAIN_MAX;

      gettext(4,3,main_width+5,main_max+6,buffer1);
      DrawHelp(main_help);
      DrawSubHelp(0,strlen(main_help)+1,sub_help1);
      display_menu(4,4,main_item,main_width+1,main_max,"MAIN MENU",7,3);

      while (TRUE)
      {
            act_line(4,4,main_item);

            switch(get_code())
            {
                    case U_ARRO:
                        del_line(4,4,main_item);
                        v_pos = (v_pos>0) ? –v_pos
                                            : main_max-1;
                        DrawSubHelp(v_pos,strlen(main_help)+1,sub_help1);
                        break;

                    case D_ARRO:
                        del_line(4,4,main_item);
                        v_pos = (v_pos<main_max-1) ?
                                            ++v_pos : 0 ;
                  DrawSubHelp(v_pos,strlen(main_help)+1,sub_help1);
                        break;

                    case '\r':
/*                      puttext(4,4,main_width+5,main_max+6,buffer1);   */
                        main_action();
                        break;
            }
    }
}

main_action()
{
char combuff[100],inputname[50];
int savebuff[80][10],namebuff[80][10];
struct text_info r;

    switch(v_pos)
    {
          case 0 :
            DrawHelp(init_help);
            setup();
            init();
            break;
 case 1 :
            DrawHelp(reset_help);
            command();
            break;
 case 2:
            DrawHelp(digit_help);
            disp_digital();
            break;
 case 3 :
            DrawHelp(ana_help);
            analog_input();
            break;
 case 4 :
            gettext(4,4,80,10,namebuff);
            draw_box(18,7,60,3);
            gotoxy(22,6);
            textcolor(11);
            cputs(" INPUT FILE NAME ");
            textcolor(15);
            gotoxy(22,7);
            cursor();
            cputs("File name : ");
            textcolor(14);
            memset(inputname,0,50);
            gets(inputname);
            gotoxy(4,4);
            nocursor();
```

```
                        puttext(4,4,80,10,namebuff);


                        if (!strlen(inputname)) break;
                         memset(combuff,0,100);
                         sprintf(combuff,"copy thesis.dat ");
                         strcat(combuff,inputname);
                         gettext(4,4,80,10,savebuff);
                         if (system(combuff) == -1) {
                                 ErrMsg("Save as error !");
                       puttext(4,4,80,10,savebuff);
                         } else {
                      puttext(4,4,80,10,savebuff);
                                 ErrMsg("1 file copy complete !");
                         }
                        nocursor();
                        break;
                case 5:
                       textbackground(0);textcolor(15);
                       puttext(4,4,19,10,buffer1);
                       puttext(1,1,80,24,buffer0);     /* restore big screen   */
                       cursor();
/*                     fclose(fp);  */
                       window(1,1,80,25);
                       clrscr();
                       exit(1);
        }
        main_menu();
}
```

```
void  command()
{
      char   *menu_item[MENU_WIDTH] =
                {" RESET                              ",
                 " WRITE ALL DIGITAL OUTPUT TO 0        ",
                 " WRITE ALL DIGITAL OUTPUT TO 1        ",
                 " READ ANALOG OUTPUT VOLTAGE           ",
                 " WRITE VOLTAGE TO ANALOG OUTPUT CARD "};

     int      menu_width = MENU_WIDTH;
     int      max_item = MAX_ITEM;
     int      act;

     gettext(4,3,menu_width+3,max_item+6,buffer3);
     gettext(2,21,55,23,buffer4);
     clrscr();
     display_menu(4,4,menu_item,menu_width,max_item,"INPUT OUTPUT COMMAND",13,3);
     while (TRUE)
     {
            act_line(4,4,menu_item);

            switch(get_code())
            {
                    case U_ARRO:
                         puttext(2,21,55,23,buffer4);
                         del_line(4,4,menu_item);
                         v_pos = (v_pos>0) ? --v_pos
                                          : max_item-1;
                         break;

                    case D_ARRO:
```

```
                    puttext(2,21,55,23,buffer4);
                    del_line(4,4,menu_item);
                    v_pos = (v_pos<max_item-1) ?
                                        ++v_pos : 0 ;
                    break;

                case '\r':
                    puttext(2,21,55,23,buffer4);
                    act = menu_action();
                    break;

                case ESC :
                    puttext(2,21,55,23,buffer4);
                    puttext(4,4,menu_width+3,max_item+6,buffer3);
                    gotoxy(4,22);clreol();
                    act = 0;
                    break;
            }
            if (!act)
            { v_pos = 0;
              break;
            }
        }
}

void  display_menu(int left, int top, char *menu_item[],int width, int max, char *header,int x, int y)
{
    int  j;

    textbackground(1);textcolor(14);
    draw_box(left,top,width,max+2);
```

```
        for( j=0; j<max; j++)
        {
                if (j==v_pos)
                   textbackground(14); textcolor(11);
                   gotoxy(x,y);
                   cputs(header);
                   textcolor(14);
                   gotoxy(left,j+top);
                   cputs(menu_item[j]);
                   textbackground(1);
        }
}

void  del_line(int left,int top, char *menu_item[])
{
        textbackground(1);
        gotoxy(left,v_pos+top);
        cputs(menu_item[v_pos]);
}

void  act_line(int left, int top, char *menu_item[])
{
        textbackground(14);
        gotoxy(left,v_pos+top);
        cputs(menu_item[v_pos]);
        textbackground(1);
}
```

```
void  draw_box(int  left, int top, int width, int height)
{
      int    j;
      char  buff[81];

      window(left,top,left+width-1,top+height-1);
      clrscr();
      window(left,top,left+width-1,top+height);

      for(j=1; j < width; j++)
             buff[j] = 196;
             buff[width] ='\0';
             buff[0] = 218;
             buff[width-1] = 191;
             gotoxy(1,1);
             cputs(buff);
             buff[0] = 192;
             buff[width-1] = 217;
             gotoxy(1,height);
             cputs(buff);

     for(j=2; j < height; j++)
     {
            gotoxy(1,j);
            putch(179);
            gotoxy(width,j);
            putch(179);
     }
     window(2,2,78,23);
}
```

```
char  get_code(void)
{
      char   key;

      if((key=getch()) == 0)
           return (getch());
      else if (key == '\r')
               return (key);
      else if (key == ESC)
               return (key);
      else
               return (0);
}

int   menu_action()
{
      int   x,i;

      switch(v_pos)
      {
                case 0 :
                         for (i=4; i<A_D_VAL[0]+4; i++)
                             A_D_VAL[i] = 255;
                         fseek(fp,0,0);
                         for (i=0; i<A_D_VAL[0]+4; i++)
                           fprintf(fp,"%s %d\n",a_d_name[i],A_D_VAL[i]);
                         sport(PORT,'R');
                         recv_port();
                         return(1);
                case 1 :
                         sport(PORT,'C');
```

```
                        recv_port();
                        return(1);
                case 2 :
                        sport(PORT,'D');
                        recv_port();
                        return(1);
                case 3:
                        sport(PORT,'u');
                        do {
                          x = card1("ENTER CARD TO BE READ (0-3) ");
                        } while (x == 0);
                        recv_port();
                        return(1);
                case 4 :
                        sport(PORT,'U');
                        do {
                          x = sub2("ENTER CARD TO BE WRITE (0-3) ");
                        } while (x == 0);
                        recv_port();
                        return(1);
        }
}

void init()
{
    char   a_d_str[14][8],typ_str[10][2];
    int    ai_lim[10];
    int    i,j;

    sport(PORT,'I');
```

```
  for (i=0; i<4; i++)
          itoa(A_D_VAL[i],a_d_str[i],10);
  for (i=0; i<A_D_VAL[0];i++)
  {   ai_lim[i] = ((A_D_VAL[i+4]+KETOCEL)/(100*0.0196));
          itoa(ai_lim[i],a_d_str[i+4],10);
  }


  for (i=0; i<A_D_VAL[0]+4; i++)
  {
        while(rport(PORT) != CR) {}


        for (j=0; j<strlen(a_d_str[i]); j++)      /* send analog and digital */
            sport(PORT,a_d_str[i][j]);            /* input, output and limit */
        sport(PORT,CR);                            /* to 8051 board */
  }
  while(rport(PORT) != CR) {}


  for (i=0; i<A_D_VAL[0]; i++)
        itoa(A_TYPE[i],typ_str[i],10);


  for (i=0; i<A_D_VAL[0]; i++)                 /*  send analog sensor type  */
  {                                           /*  to 8051 board            */
        while(rport(PORT) != CR) {}
        for (j=0; j<strlen(typ_str[i]); j++)
            sport(PORT,typ_str[i][j]);
        sport(PORT,CR);
  }
  while(rport(PORT) != CR) {}
}
```

```
void  setup_display(int left, int top, char *menu_item[],int width,int max, char *header,int x, int y)
{
      int  j;


      textbackground(1);textcolor(14);
      draw_box(left,top,width,max+2);


      for( j=0; j<max; j++)
      {
             if (j==v_pos)
                textbackground(1); textcolor(11);
                gotoxy(x,y);
                cputs(header);
                textbackground(1); textcolor(14);
            gotoxy(left,j+top);
                cputs(menu_item[j]);
                printf("%d",A_D_VAL[j]);
      }
}

void  setup()
{
      char  *menu2[35] =
             {" Number of analog input sensor : ",
              " Number of analog output card  : ",
              " Number of digital input port  : ",
              " Number of digital output port : "};


      int  act,i;
      char  ch;
```

```
A_D_VAL[2] = 12 - A_D_VAL[3];


gettext(4,3,50,20,buffer2);


while (TRUE)
{
        cursor();
        clrscr();
        A_D_VAL[3] = 12 - A_D_VAL[2];
        setup_display(4,3,menu2,39,4," SET UP ",18,2);
        gotoxy(37,v_pos+3);
        printf("%d",A_D_VAL[v_pos]);
        gotoxy(37,v_pos+3);
        switch(get_code())
        {
                case U_ARRO:
                    v_pos = (v_pos>0) ? –v_pos
                                                : 3;
                    act = 1;
                    break;
/*              case CR : */
                case D_ARRO:
                    v_pos = (v_pos < 3) ?
                                                ++v_pos : 0 ;
                    act = 1;
                    break;
                case F1 :
                    act = ad_init();
                    v_pos = (v_pos < 3) ?
                                                ++v_pos : 0 ;
                    act = 1;
```

```
                        break;


                      case ESC :
                     analog_setup();
                        fclose(fp);
                        fp = fopen("THESIS.CFG","w");
                        for (i=0; i<A_D_VAL[0]+4; i++)
                          fprintf(fp,"%s %d\n",a_d_name[i],A_D_VAL[i]);

                        for (i=0; i<A_D_VAL[0]; i++)
                          fprintf(fp,"%s %d\n",ana_type[i],A_TYPE[i]);

                        act = 0;
                        break;
                }

                if (!act)
                { v_pos = 0;

                  nocursor();
                  puttext(4,3,50,20,buffer2);
                  fclose(fp);
                  break;
                }
         }
     }
```

```
void numinput(int len,int *intbuff) {
char ch, strbuff[5];
int xlen;

        memset(strbuff,0,5);
        xlen=0;
        do {
        ch = getch();
        if ((ch >= 48) && (ch <= 57)) {
        if (strlen(strbuff) < len) {
                strbuff[xlen] = ch;
                putch(ch);
                xlen++;
        } else {

        }
        } else {
                if (ch == '\b') {
                        if (xlen >=1) {
                        strbuff[–xlen] = '\0';
                        gotoxy(wherex()-1,wherey());
                        putch(' ');
                gotoxy(wherex()-1,wherey());
                        }
                }
        }
        } while(ch != CR);
        *intbuff = atoi(strbuff);
}
```

```
int ad_init()
{
   int i,QUIT =0;
   char adbuff[3];

   gotoxy(37,3+v_pos);
   cputs("   ");
   gotoxy(37,3+v_pos);
   do {
          numinput(2,&A_D_VAL[v_pos]);
          switch (v_pos) {
                    case 0 : if ((A_D_VAL[v_pos] > 8) || (A_D_VAL[v_pos] <= 0)) {
                                 QUIT = 0;
                        gotoxy(37,3+v_pos);
                                 cputs("   ");
                        ErrMsg("MAXIMUM SENSOR MUST BE BETWEEN 0 - 8 !");
                                 gotoxy(37,3+v_pos);
                               } else {
                                 QUIT = 1;
                               }
                              break;
                    case 1 : if ((A_D_VAL[v_pos] > 8) || (A_D_VAL[v_pos] <= 0)) {
                                 QUIT = 0;
                        gotoxy(37,3+v_pos);
                                 cputs("   ");
                                 ErrMsg("MAXIMUM ANALOG OUTPUT MUST BE BETWEEN 0 - 8 !");
                                 gotoxy(37,3+v_pos);
                               } else {
                                 QUIT = 1;
                               }
                              break;
```

```
                case 2 : if ((A_D_VAL[v_pos] > 12) || (A_D_VAL[v_pos] <= 0)) {
                            if ((A_D_VAL[v_pos]+A_D_VAL[3]) > 12 ) {
                            QUIT = 0;
                  gotoxy(37,3+v_pos);
                            cputs("   ");
                            ErrMsg("MAXIMUM DIGITAL INPUT MUST BE BETWEEN 0 - 12 !");
                            gotoxy(37,3+v_pos);
                            }
                          } else {
                            QUIT = 1;
                          }
                        break;
               case 3 : if ((A_D_VAL[v_pos] > 12) || (A_D_VAL[v_pos] <= 0)) {
                            if ((A_D_VAL[v_pos]+A_D_VAL[2]) > 12 ) {
                            QUIT = 0;
                  gotoxy(37,3+v_pos);
                            cputs("   ");
                            gotoxy(37,3+v_pos);
                            }
                          } else {
                            QUIT = 1;
                          }
                        break;
        }
    } while (!QUIT);

    return(1);
}
```

```
recv_port()
{
      char  ch ;

      gettext(2,21,55,23,buffer4);
      draw_box(2,21,53,3);
      textcolor(11);
      gotoxy(4,20);
      cputs(" REMOTE RESPONSE ");
      textcolor(14);
      gotoxy(4,21);

      while ((ch = rport(PORT)) != CR)
               putchar(ch);
}

analog_setup()
{
      int   d_buff[80][25];
      int   x,y,act,ans;
      int   lim_pos = 4;

      x = 28;y = 6;

      gettext(15,3,79,18,d_buff);
      draw_box(23,3,50,A_D_VAL[0]+6);
      nocursor();              textcolor(11);
      gotoxy(40,2);
      cputs(" ANALOG SETUP ");
      gotoxy(24,4);
      textcolor(15);
```

```
cputs("  SENSOR      TYPE       LIMIT       UNIT ");
textcolor(14);
DrawHelp(ana_help);
analog_setup_screen(y);
while (TRUE)
{
     gotoxy(x,y);
     switch (get_code())
     {
          case U_ARRO :
                    y--;
                    if (y<6)
                       y=A_D_VAL[0]+5;

                    lim_pos--;
                    if (lim_pos < 4)
                      lim_pos = A_D_VAL[0]+3;
                    analog_setup_screen(y);
                    act = 1;
                    break;

          case D_ARRO :
                    y++;
                    if (y > A_D_VAL[0]+5)
                       y = 6;

                    lim_pos++;
                    if (lim_pos > A_D_VAL[0]+3)
                       lim_pos = 4;
                    analog_setup_screen(y);
                    act = 1;
```

```
                        break;

                    case F1:
                        act = analog_col_setup(x,y,lim_pos);
                        break;

                    case ESC :
                        act = 0;
                        break;
                }
                if (!act)
                {  puttext(15,3,79,18,d_buff);
                   break;
                }
            }
        }

analog_setup_screen(int y_pos)
{
        int   d_buff[80][25];
        int   i,j;
        int   len = 0;
        char  val_str[5];

        for (i=0; i<A_D_VAL[0]; i++)
        {
            if(i == y_pos-6)
              textbackground(14);
            else
              textbackground(1);
            gotoxy(26,i+6);
```

```
  cputs("  "); putch(0x31+i); cputs("        ");
  gotoxy(35,i+6);
  switch(A_TYPE[i])
  {
    case 0 :
             cputs("Temperature     ");
             gotoxy(60,i+6);
             cputs("Celcias ");
             break;
    case 1 :
             cputs("Resistance      ");
             gotoxy(60,i+6);
             cputs("Ohms     ");
             break;
    case 2 :
             cputs("Humidity        ");
             gotoxy(60,i+6);
             cputs("Percent ");
             break;
    case 3 :
             cputs("Voltage         ");
             gotoxy(60,i+6);
             cputs("Volts   ");
             break;
    case 4 :
             cputs("Speed           ");
             gotoxy(60,i+6);
             cputs("Rpm     ");
             break;
  }
itoa(A_D_VAL[i+4],val_str,10);
```

```
            gotoxy(50,i+6);
                cputs(val_str);
                len = strlen(val_str);
                for(j=0; j<(6+(4-len)); j++)
                  cputs(" ");
                textbackground(1);
        }
}

int analog_col_setup(int x, int y, int lim_pos)
{
        int act,i,QUIT =0;
        int a = 0; int c = 0;

        textbackground(14);
        while (TRUE)
        {
              gotoxy(x+7,y);
              cursor();
              switch (get_code())
              {
                   case R_ARRO :
                             x+=15;
                             if (x>50)
                                  x=28;
                             a++;
                             if (a > 1)
                                  a = 0;
                             act = 1;
                             break;
```

```
case L_ARRO :
        x-=15;
        if (x < 25)
            x = 43;
        a--;
        if (a < 0)
            a = 1;
        act = 1;
        break;

case F1:
        if (a > 0)
         {
            cputs("    ");
            gotoxy(x+7,y);
            do {
                   numinput(3,&A_D_VAL[lim_pos]);
                   if((A_D_VAL[lim_pos] > 255) || (A_D_VAL[lim_pos] <= 0))
                    {
                      QUIT = 0;
                      gotoxy(x+7,y);
                      cputs("    ");
                      gotoxy(x+7,y);
                    }
                    else
                      {
                           QUIT = 1;
                      }
                  } while (!QUIT);
         }
```

```
else
   while ( getch() != CR)
   {
         nocursor();
         if (c > 4)
             c = 0;
         A_TYPE[lim_pos-4] = c;
         gotoxy(x+6,y);
         cputs(type_item[c]);
         gotoxy(x+32,y);cputs("          ");
         gotoxy(x+32,y);

         switch(c)
         {
            case 0:
                     cputs("Celcias");
                     break;
            case 1:
                     cputs("Ohms");
                     break;
            case 2:
                     cputs("Percent");
                     break;
            case 3:
                     cputs("Volts");
                     break;
            case 4:
                     cputs("Rpm");
         }
         c++;
}
```

```
                                        cursor();
                                        break;
                                case ESC :
                                        act = 0;
                                        nocursor();
                                        break;
                        }
                        if (!act)
                            break;
                }
                return (1);
        }

        analog_input()
        {
                char wbuff[50],ch,xch[2];
                int   i = 0;
                int   n,ferr = 0;
                char  lim_val[10],di_val[10],max_val[10],t_char[2];
                int   a_buff[80][25];
                FILE *fptr;
                struct time timep;
                struct date datep;

                gettext(4,3,79,10+A_D_VAL[0],a_buff);
                draw_box(4,3,73,6+A_D_VAL[0]);
                textcolor(11);
                gotoxy(30,2);
                cputs(" ANALOG INPUT ");
                textcolor(14);
                gotoxy(6,4);
```

```
textcolor(15);
cputs("SENSOR TYPE     LIMIT      VALUE      MAX      UNIT         STATUS");
textcolor(14);
if ((fptr = fopen(FILEBUFF,"w")) == NULL )
{
      ErrMsg("Open buffer file error !");
      ferr =1;
}
else
      ferr =0;

getdate(&datep);
fprintf(fptr,"%d\/%d\/%d\n",datep.da_day,datep.da_mon,datep.da_year);
gettime(&timep);
fprintf(fptr,"%d:%d:%d\n\n",timep.ti_hour,timep.ti_min,timep.ti_sec);
fprintf(fptr,"SENSOR_NO SENSOR_TYPE\tLIMIT\tVALUE\tMAX\tSTATUS\n\n");


while(!kbhit())
{
      sport(PORT,'V');

      for (n=0; n<A_D_VAL[0]; n++)
      {
          while(rport(PORT) != CR) {}
          delay(100);
          sport(PORT,'T');
      while(rport(PORT) != CR) {}
          itoa(A_TYPE[n],t_char,10);
          sport(PORT,t_char[0]);
      while(rport(PORT) != CR) {}
```

```
memset(wbuff,0,50);
gotoxy(6,6+n);

switch(A_TYPE[n])
{
        case 0 :
                 cputs("Temperature");
                 gotoxy(52,6+n);
                 cputs("Celcias");
                 break;
        case 1 :
                 cputs("Resistance");
                 gotoxy(52,6+n);
                 cputs("Ohms");
                 break;
        case 2 :
                 cputs("Humidity");
                 gotoxy(52,6+n);
                 cputs("Percent");
                 break;
        case 3 :
                 cputs("Voltage");
                 gotoxy(52,6+n);
                 cputs("Volts");
                 break;
        case 4 :
                 cputs("Speed");
                 gotoxy(52,6+n);
                 cputs("Rpm");
                 break;
}
```

```
 sprintf(wbuff,"%d %d\t",n,A_TYPE[n]);

 while(rport(PORT) != CR) {}
 delay(100);
 sport(PORT,'W');
 gotoxy(64,6+n);

 while ((ch = rport(PORT)) != CR)
 {
        if (ch == 'N')
        {   textcolor(12);
            cputs("Over Limit");
            textcolor(14);
        }
        else if(ch == 'Y')
             cputs("           ");
}

delay(100);
sport(PORT,'L');

while(rport(PORT) != CR) {}
i=0;

while ((ch = rport(PORT)) != CR)
{
           gotoxy(21+i,6+n);
           lim_val[i] = ch;
           putch(ch);
           xch[0] =ch;xch[1] = '\0';
           strcat(wbuff,xch);
```

```
                i++;
        }
        strcat(wbuff,"\t");
        delay(100);
        sport(PORT,'V');
        while(rport(PORT) != CR) {}
        i = 0;

        while ((ch = rport(PORT)) != CR)
        {
                gotoxy(32+i,6+n);
                di_val[i] = ch;
                putch(ch);
                 xch[0] =ch;xch[1] = '\0';
                 strcat(wbuff,xch);
                i++;
        }

strcat(wbuff,"\t");
        delay(100);
        sport(PORT,'M');
        while(rport(PORT) != CR) {}
        i = 0;

        while ((ch = rport(PORT)) != CR)
        {
                gotoxy(42+i,6+n);
                max_val[i] = ch;
                putch(ch);
                 xch[0] =ch;xch[1] = '\0';
                 strcat(wbuff,xch);
```

```
                    i++;
                }

                if (!ferr)
                        fprintf(fptr,"%s\n",wbuff);
            }
        }
        puttext(4,3,79,10+A_D_VAL[0],a_buff);
        v_pos = 0;

        getdate(&datep);
        fprintf(fptr,"\n%d\/%d\/%d\n",datep.da_day,datep.da_mon,datep.da_year);
        gettime(&timep);
        fprintf(fptr,"%d:%d:%d\n",timep.ti_hour,timep.ti_min,timep.ti_sec);
        fclose(fptr);
}

void  sport(port,c)
int   port;
char  c;
{
   union  REGS  r;

   r.x.dx  = port;
   r.h.al  = c;
   r.h.ah  = 1;
   int86(0x14,&r,&r);
   if (r.h.ah & 128)
   {
      ErrMsg("send error detected in serial port ");
      exit(1);
```

```
    }
}

int card1(char *st)
{
      char  ch[4];
      char  buffer[80][25];
      int   i = 0;
      int   num;



     cursor();
     gettext(34,6,70,9,buffer);
     textbackground(1);textcolor(14);
     draw_box(34,6,35,3);
     gotoxy(36,6);
     cputs(st);
     scanf("%s",&ch);

     num = atoi(ch);
     if (num > A_D_VAL[1]-1)
     {  puttext(34,6,70,9,buffer);
           return(0);            /*  return 0 if card > the card installed */
     }
     do {
            sport(PORT,ch[i]);
            i++;
     } while (i < strlen(ch));
     sport(PORT,CR);
     nocursor();
     puttext(34,6,70,9,buffer);
    return(1);
}

int  sub2(char *st)
{
    char  port_no[4],volt[8];
    char  sub2_buf[80][25];
    int   num;
    int   i = 0;

    cursor();
    gettext(34,6,70,10,sub2_buf);
    textbackground(1);textcolor(14);
    draw_box(34,6,35,4);
    gotoxy(36,6);
    cputs(st);
    scanf("%s",&port_no);

    num = atoi(port_no);
    if (num > A_D_VAL[1]-1)
    {  puttext(34,6,70,9,sub2_buf);
          return(0);          /*   return 0 if card exceed the card installed   */
    }
    gotoxy(36,7);
    cputs("VALUE : ");
    i=0;
    scanf("%s",&volt);

    do {
          sport(PORT,port_no[i]);
          i++;
```

```
        } while (i < strlen(port_no));
        sport(PORT,CR);

        i=0;
        do {
                sport(PORT,volt[i]);
                i++;
        } while (i < strlen(volt));
        sport(PORT,CR);
        puttext(34,6,70,10,sub2_buf);
        nocursor();
        return(1);
}

void ErrMsg(char *str) {
int     err_buff[80][25];

        gettext(4,20,65,25,err_buff);
        draw_box(4,20,strlen(str)+5,3);
        gotoxy(6,19);
        textcolor(11);
        cputs(" MESSAGE ");
        textcolor(15);
        gotoxy(6,20);
        nocursor();
        cputs(str);
        textcolor(14);
        getch();
        cursor();
        puttext(4,20,65,25,err_buff);
}
```

```
rport(port)
int  port;
{
   int     err_buff[80][25];
   union  REGS  r;


   r.x.dx  = port;
   r.h.ah  = 2;
   int86(0x14,&r,&r);
   if (r.h.ah & 128)
   {  gettext(4,20,65,25,err_buff);
      draw_box(4,20,45,4);
      gotoxy(6,19);
      textcolor(11);
      cputs(" ERROR ");
      textcolor(15);
      gotoxy(6,20);
      cputs(" Read error detected in serial port.");
      gotoxy(6,21);
      cputs(" Check communication line or 8051 board.");
      textcolor(14);
      getch();
      puttext(4,20,65,25,err_buff);
   }
   return r.h.al;
}
```

```
 check_stat(port)
 int  port;
 {
    union  REGS  r;

    r.x.dx  = port;
    r.h.ah  = 3;
    int86(0x14,&r,&r);
    return r.x.ax;
 }

void  port_init(port)
int   port;
{
   union  REGS  r;

   r.x.dx  = port;
   r.h.ah  = 0;
   r.h.al  = 231;
   int86(0x14,&r,&r);
}

void nocursor()
{
   _AH = 1;
   _CH = 32;
   geninterrupt(0x10);
}
```

```
void cursor()
{
    _AH = 1;
    _CH = 3;
    _CL = 4;
    geninterrupt(0x10);
}

void alarm()
{
  while (!getche())
  {  sound(1400);
     delay(1000);
     sound(300);
     delay(1500);
  }
  nosound();
}

disp_digital()
{
      int x, y, act, port,oport;
      int   scr_buff[80][25];

      gettext(0,0,79,24,scr_buff);
      clrscr();

      act = 1; port=1; oport = 12;
      d_port_scr(port);
      x = 7; y = 3;
      gotoxy(x,y);
```

```
while (TRUE)
{
      switch (get_code())
      {
       case R_ARRO :
         oport = port;
             port++;
             if (port > 12)
                    port = 1;
             gotoxy(x,y);
             act = 1;
             d_port_newscr(port,oport,1);
             break;

      case L_ARRO :
            oport = port;
            port--;
            if (port < 1)
                   port = 12;
            gotoxy(x,y);
            act = 1;
            d_port_newscr(port,oport,1);
            break;

     case F1:
           if (port > A_D_VAL[2])             /* do nothing if port = INPUT */
                  act = out_port(port-1,port,oport);
           break;

     case ESC :
           puttext(0,0,79,24,scr_buff);
```

```
                        act = 0;
                        break;
                }
                if (!act)
                        break;
        }
}

read_i_port(int port)
{
    char  ch,ip_str[4];
    int   i=0;

    itoa(port,ip_str,10);

    sport(PORT,'f');
    delay(100);
    do {
        sport(PORT,ip_str[i]);
        i++;
    } while (i < strlen(ip_str));
    sport(PORT,CR);
    sport(PORT,CR);

    while (rport(PORT) != CR)  {}

    while ((ch = rport(PORT)) != CR)
            putch(ch);
}
```

```
 read_o_port(int port)
 {
    char    op_str[3],ch;
    int     i = 0;

             itoa(port,op_str,10);
             delay(100);
             sport(PORT,'b');
             delay(100);
             do {
                 sport(PORT,op_str[i]);
                 i++;
             } while (i < strlen(op_str));
             sport(PORT,CR);

             while(rport(PORT) != CR)  {}
             while ((ch = rport(PORT)) != CR)
                   putch(ch);
}

int out_port(int port, int x, int y)
{
    char  p_str[4],value[5],ch;
    int   i,xlen,QUIT=0;
    int   outbuff;
    textcolor(12);

    itoa(port,p_str,10);

    cursor();
    d_port_newscr(x,y,0);
```

```
do {
       memset(value,0,5);
       xlen=0;
       do {


           ch = getch();
           if ((ch >= 48) && (ch <= 57))
           {
              if (strlen(value) < 3)
              {
                     value[xlen] = ch;
                     putch(ch);
                     xlen++;
              }
           }
           else
           {
                  if (ch == '\b')
                  {
                     if (xlen >=1)
                     {
                        value[–xlen] = '\0';
                        gotoxy(wherex()-1,wherey());
                        putch(' ');
                        gotoxy(wherex()-1,wherey());
                     }
                  }
           }
        } while(ch != CR);

           outbuff = atoi(value);
```

```
    if ((outbuff > 255) II (outbuff <= 0))
    {
           QUIT = 0;
           gotoxy(wherex()-3,wherey());
           cputs("   ");
           gotoxy(wherex()-3,wherey());
    }
    else
     {
           QUIT = 1;
     }
} while (!QUIT);

 sport(PORT,'B');
 delay(100);
 i=0;

 do {
          sport(PORT,p_str[i]);
          i++;
    } while( i < strlen(p_str));
 sport(PORT,',');

 i=0;
 do {
          sport(PORT,value[i]);
          i++;
    } while ( i < strlen(value));
 sport(PORT,CR);

 while(rport(PORT) != CR)  {}
```

```
            d_port_newscr(x,y,1);
            nocursor();
            return(1);
}

void d_port_scr(int port)
{
    int   i,j,k,l;
    char  ch[3], di_buff[81];
    int   port_no = 0;
    char  c;

    textbackground(1);

    for (k=0; k<15; k+=7)
    {
        for (i=0; i<61; i+=19)
        {
          itoa(port_no,ch,10);
          port_no++;

          if (port_no == port)
             textcolor(12);
          else textcolor(14);
          for(j=1; j < 17; j++)
             di_buff[j] = 196;
          di_buff[17] ='\0'; di_buff[0] = 218;  di_buff[16] = 191;

          gotoxy(2+i,2+k);  cputs(di_buff);

          di_buff[0] = 195;  di_buff[16] = 180;
```

```
            gotoxy(2+i,3+k);  putch(179);
            gotoxy(18+i,3+k);  putch(179);
            gotoxy(8+i,3+k);
            cputs("PORT "); cputs(ch);
            gotoxy(2+i,4+k);  cputs(di_buff);

            for(j=2; j < 17; j+=2)
               di_buff[j] = 196;
            di_buff[0] = 192;  di_buff[16] = 217;

            gotoxy(2+i,7+k);  cputs(di_buff);
            gotoxy(2+i,5+k);  putch(179);
            gotoxy(3+i,5+k);  cputs("Status : ");
            if (A_D_VAL[2] == 0)
               cputs("OUTPUT");
            else  if (port_no <= A_D_VAL[2])
                          cputs("INPUT");
                     else  cputs("OUTPUT");
            gotoxy(18+i,5+k); putch(179);
            gotoxy(2+i,6+k);  putch(179);
            gotoxy(18+i,6+k); putch(179);
            gotoxy(3+i,6+k);  cputs("Value : ");

            if (port_no <= A_D_VAL[2])
                read_i_port(port_no-1);
            else  read_o_port(port_no-1);
            textcolor(14);
         }
      }
   }
```

```
void d_port_newscr(int port,int oport,int flag)
{
    int   i,j,k,l;
    char  ch[3], di_buff[81];
    int   port_no = 0;
    char  c;

    textbackground(1);
    for (k=0; k<15; k+=7)
    {
        for (i=0; i<61; i+=19)
        {
          itoa(port_no,ch,10);
          port_no++;
          if (port_no != port)    continue;
          d_port_oldscr(oport);
          if (port_no == port)
             textcolor(12);
          else textcolor(14);
          for(j=1; j < 17; j++)
             di_buff[j] = 196;
          di_buff[17] ='\0'; di_buff[0] = 218;  di_buff[16] = 191;

          gotoxy(2+i,2+k);  cputs(di_buff);

          di_buff[0] = 195;  di_buff[16] = 180;
          gotoxy(2+i,3+k);  putch(179);
          gotoxy(18+i,3+k);  putch(179);
          gotoxy(8+i,3+k);
          cputs("PORT "); cputs(ch);
          gotoxy(2+i,4+k);  cputs(di_buff);
        for(j=2; j < 17; j+=2)
           di_buff[j] = 196;
        di_buff[0] = 192;  di_buff[16] = 217;


        gotoxy(2+i,7+k);  cputs(di_buff);
        gotoxy(2+i,5+k);  putch(179);
        gotoxy(3+i,5+k);  cputs("Status : ");
        if (A_D_VAL[2] == 0)
           cputs("OUTPUT");


        else  if (port_no <= A_D_VAL[2])
                      cputs("INPUT");
                 else  cputs("OUTPUT");
        gotoxy(18+i,5+k); putch(179);
        gotoxy(2+i,6+k);  putch(179);
     gotoxy(18+i,6+k); putch(179);
        gotoxy(3+i,6+k);  cputs("Value : ");
        if (port_no <= A_D_VAL[2])
           read_i_port(port_no-1);
        else if (flag != 0)
                    read_o_port(port_no-1);
        else {
                 gotoxy(3+i,6+k);  cputs("Value :      ");
                 gotoxy(3+i+8,6+k);
                }
        textcolor(14);
        break;
     }
  }


}
```

```
void d_port_oldscr(int oport)
{
     int   i,j,k,l;
     char  ch[3], di_buff[81];
     int   port_no = 0;
     char  c;

     textbackground(1);

     for (k=0; k<15; k+=7)
     {
          for (i=0; i<61; i+=19)
          {
             itoa(port_no,ch,10);
             port_no++;
             if (port_no != oport) continue;
             if (port_no == oport)
                textcolor(14);
             else textcolor(12);
             for(j=1; j < 17; j++)
                di_buff[j] = 196;
             di_buff[17] ='\0'; di_buff[0] = 218;  di_buff[16] = 191;

             gotoxy(2+i,2+k);  cputs(di_buff);

             di_buff[0] = 195;  di_buff[16] = 180;
             gotoxy(2+i,3+k);  putch(179);
             gotoxy(18+i,3+k);  putch(179);
             gotoxy(8+i,3+k);
             cputs("PORT "); cputs(ch);
             gotoxy(2+i,4+k);  cputs(di_buff);
```

```
                  for(j=2; j < 17; j+=2)
                     di_buff[j] = 196;
                  di_buff[0] = 192;  di_buff[16] = 217;

                  gotoxy(2+i,7+k);  cputs(di_buff);
                  gotoxy(2+i,5+k);  putch(179);
                  gotoxy(3+i,5+k);  cputs("Status : ");
                  if (A_D_VAL[2] == 0)
                     cputs("OUTPUT");
                  else  if (port_no <= A_D_VAL[2])
                              cputs("INPUT");
                        else  cputs("OUTPUT");
                  gotoxy(18+i,5+k); putch(179);
                  gotoxy(2+i,6+k);  putch(179);
               gotoxy(18+i,6+k); putch(179);
                  gotoxy(3+i,6+k);  cputs("Value : ");

                  if (port_no <= A_D_VAL[2])
                     read_i_port(port_no-1);
                  else  read_o_port(port_no-1);
                  textcolor(14);
                  break;
               }
         }
}

void DrawHelp(char *help_buffer)
{
        struct text_info r;

        gettextinfo(&r);
        window(1,1,80,25);
        gotoxy(1,25);
        textbackground(7);
        textcolor(0);
        cputs(help_buffer);
        textbackground(1);
        textcolor(14);
        window(r.winleft,r.wintop,r.winright,r.winbottom);
}

void DrawSubHelp(int cur,int x,char *subhelp[])
{
        struct text_info r;

        gettextinfo(&r);
        window(1,1,80,25);
        gotoxy(x,25);
        textbackground(7);
        textcolor(4);
        cputs(subhelp[cur]);
        textbackground(1);
        textcolor(14);
        window(r.winleft,r.wintop,r.winright,r.winbottom);
}
~
```

## ประวัติผู้เขียน

นายปกรณ์ ชุณหสวัสดิกุล เกิดวันที่ 10 กุมภาพันธ์ 2509 จังหวัดขอนแก่น สำเร็จการศึกษาปริญญาตรีอุตสาหกรรมศาสตร์บัณฑิต สาขาเทคโนโลยีคอมพิวเตอร์-อุตสาหกรรม ภาควิชาวัดคุมทางอุตสาหกรรม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง ในปีพ.ศ. 2532 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตร์คอมพิวเตอร์ ที่จุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ. 2535

ปัจจุบัน ทำงานในตำแหน่งผู้ประสานงานข้อมูลสาระสนเทศ ( IT Coordinator ) บริษัท 3 เอ็ม (ประเทศไทย) จำกัด