```
DATAOUT      EQU 0FFF0H   ;데이터 아웃의주소
DATAIN       EQU 0FFF1H   ;데이터 인의주소


RWKEY        EQU 10H ;READ AND WRITE KEY
INCKEY       EQU 11H ;INCRESE KEY(COMMA ,)
ENDKEY       EQU 12H ;END KEY (PERIOD . )
GO           EQU 13H ;GO-KEY
REG          EQU 14H ;REGISTER KEY
DECKEY       EQU 15H ;DECRESE KEY
CODE         EQU 16H ;CODE KEY
ST           EQU 17H ;SINGLE STEP KEY
RST          EQU 18H ;RST KEY


SEG1   EQU   0FFC3H
SEG2   EQU   0FFC2H
SEG3   EQU   0FFC1H

SCORE      EQU 31H




LCDWIR       EQU 0FFE0H   ;LCD IR 쓰기
LCDWDR       EQU 0FFE1H   ;LCD DR 쓰기
LCDRIR       EQU 0FFE2H   ;LCD IR 읽기
LCDRDR       EQU 0FFE3H   ;LCD DR 읽기

INST       EQU 20H       ;LCD INSTRUCTION 값 보관
DATA       EQU 21H       ;LCD DATA 값 보관
LROW       EQU 22H       ;LCD 표시 좌표: 행의 값 보관
LCOL       EQU 23H       ;LCD 표시 좌표: 열의 값 보관
NUMFONT       EQU 24H       ;message 개수 보관
FDPL       EQU 25H       ;DPL 값 보관
FDPH       EQU 26H       ;DPH 값 보관


SROW1       EQU 32H
SCOL1       EQU 33H
```

```
FOOD1     EQU 35H
FOOD2       EQU 36H
INPUT       EQU 39H
TIME       EQU 30H

CLEAR       EQU 01H       ;CLEAR 명령
CUR_HOME    EQU 02H       ;CURSOR HOME 위치로 이동
; 커서의 진행 방향을 제어하고,표시의 이동을 제어
ENTRY2       EQU 06H       ;어드레스를 1 증가 시키고, 커서나 블링크를 우로 이동
; 표시부 ON/OFF 제어
DCB6       EQU 0EH       ;표시(ON),커서(ON),블링크(OFF)
; Function setting
FUN5       EQU 38H       ;8비트 2행 5*7 1/16듀티
; DDRAM Address setting
LINE_1        EQU 80H       ;1 000 0000 : LCD 1 번째 줄로이동
LINE_2       EQU 0C0H    ;1 010 0000 : LCD 2 번째 줄로이동

COLGREEN    EQU 0FFC5H
COLRED       EQU 0FFC6H
ROW       EQU 0FFC7H
ORG 8000H

   MOV    SP, #60H

   MOV      TMOD,#00000001B
   MOV      IE,#10000011B

   MOV      TH0,#00H
   MOV      TL0,#00H
   MOV    TIME, #00H
   MOV    R6, #60




;LCD 초기화 수행
LCD_INIT:    MOV       INST,#FUN5
             CALL      INSTWR

             MOV       INST,#DCB6
```

```
            CALL    INSTWR

            MOV     INST,#CLEAR
            CALL    INSTWR

            MOV     INST,#ENTRY2
            CALL    INSTWR

INIT:



LCD0:   MOV     LROW,#01H
            MOV     LCOL,#04H
            CALL    CUR_MOV

            MOV     DPTR,#MESSAGE1
            MOV     FDPL,DPL
            MOV     FDPH,DPH
            MOV     NUMFONT,#0EH
            CALL    DISFONT


            MOV     LROW,#02H
            MOV     LCOL,#06H
            CALL    CUR_MOV

            MOV     DPTR,#MESSAGE2
            MOV     FDPL,DPL
            MOV     FDPH,DPH
            MOV     NUMFONT,#12H
            CALL    DISFONT

   MOV   TIME, #00H

   MOV   SCORE, #00H
   MOV   A, SCORE
   CALL   SEG

   CALL   DOTCOLG0
   CALL   DOTCOLR0
```

```
MOV    SROW1, #00010000B
MOV    SCOL1, #00001000B
CALL    DOTCOLG



CLR      C
MOV    INPUT #00H

CALL    SCANN0

MOV TH0, #00010000B
MOV TL0, #00000000B


    MOV      LROW,#01H
         MOV      LCOL,#00H
         CALL     CUR_MOV
         MOV      DPTR,#MESSAGEN
         MOV      FDPL,DPL
         MOV      FDPH,DPH
         MOV      NUMFONT,#15H
         CALL     DISFONT

         MOV      LROW,#02H
         MOV      LCOL,#01H
         CALL     CUR_MOV
         MOV      DPTR,#MESSAGE0
         MOV      FDPL,DPL
         MOV      FDPH,DPH
         CALL     DISFONTT
         MOV      DPTR,#MESSAGEN
         MOV      FDPL,DPL
         MOV      FDPH,DPH
         MOV      NUMFONT,#15H
         CALL     DISFONT

SETB    TCON.TR0


NOP
```

```
CLR   TCON.TR0
   MOV A,TL0
   MOV B,#03H
   MUL AB
   MOV R1,A
   MOV A,#00000001B
DCOL0:
   MOV R3,A
   MOV FOOD2,A
   RL A
   DJNZ R1,DCOL0
   MOV A, #00000001B
DROW0:   MOV R7,A
   MOV FOOD1,A
   RL A
   DJNZ TL0,DROW0
   MOV A,#00000001B
   CALL     DOTCOLG0
   CALL     DOTCOLR
   CALL     DELAY
SETB   TCON.TR0

MAIN:
   CALL   MOVING
   CALL   FOODING

   CALL   DOTCOLR0
   CALL   DOTCOLG
   CALL   DELAY

   CALL   DOTCOLG0
   CALL   DOTCOLR
   CALL   DELAY


   CALL   SCANN1
   JMP    MAIN



SCANN0:      PUSH    PSW      ; PSW 값을 스택에 보관
```

```
KEYINIT0:   MOV     R1,#00H          ; R1의 초기화
            MOV     A,#11101111B     ; 데이터 아웃의 초기값

COLSCAN0:   MOV     R0,A             ; R0에 데이터 아웃 값 보관
            INC     R1               ; R1 열의 값 보관
            CALL    SUBKEY           ; 키 패드 입력 상태 조사
            ANL     A,#00011111B     ; 상위 3비트 제거
            XRL     A,#00011111B     ; XRL 연산000111111
            JNZ     RSCAN0           ; 누산기 값이 0 이 아니면, 행 스캔
            MOV     A,R0
            SETB    C
            RRC     A                ; 다음 열로 이동
            JNC     KEYINIT0         ; 모든 열을 스캔했으면, 다시 시작
            JMP     COLSCAN0         ; 다음 열의 스캔을 위한 분기



RSCAN0:     MOV     R2,#00H          ; R2 행의 값 보관
ROWSCAN0:   RRC     A                ; 어느 행이 "1" 로 바뀌었는지 조사
            JC      MATRIX0          ; 캐리가 발생하면, MATRIX로 분기
            INC     R2               ; 캐리가 발생하지 않으면, 다음 행으로 이동
            JMP     ROWSCAN0         ; 다음 행의 스캔을 위한 분기

MATRIX0:    MOV     A,R2             ; R2 에는 행의 값 보존
            MOV     B,#05H           ; 1행은 5열로 이루어짐
            MUL     AB               ; 2차원 배열을 1차원 배열로 값을 바꿈
            ADD     A,R1             ; R1 에는 열의 값 보존
            CALL    INDEX            ; 키 코드 값을 지정
            CALL    ENTER            ; 입력된 값을 메모리에 저장
            POP     PSW              ; 스택으로 부터 PSW값을 가지고 옴
            RET                      ; 상위 루틴으로 복귀


SUBKEY:     MOV     DPTR, #DATAOUT
            MOVX    @DPTR, A
            MOV     DPTR, #DATAIN
            MOVX    A, @DPTR
            RET
INDEX:      MOVC    A,@A+PC          ;A IS FROM 1 TO 24
            RET
```

```
KEYBASE:   DB ST            ;SW1,ST                 1
           DB CODE          ;SW6,CODE               2
           DB DECKEY        ;SW11,CD                3
           DB REG           ;SW15,REG               4
           DB GO            ;SW19,GO                5
           DB 0CH           ;SW2,C                  6
           DB 0DH           ;SW7,D                  7
           DB 0EH           ;SW12,E                 8
           DB 0FH           ;SW16,F                 9
           DB INCKEY        ;SW20,COMMA (,)         10
           DB 08H           ;SW3,8                  11
           DB 09H           ;SW8,9                  12
           DB 0AH           ;SW13,A                 13
           DB 0BH           ;SW17,B                 14
           DB ENDKEY        ;SW21,PERIOD(.)         15
           DB 04H           ;SW4,4                  16
           DB 05H           ;SW9,5                  17
           DB 06H           ;SW14,6                 18
           DB 07H           ;SW18,7                 19
           DB RWKEY         ;SW22,R/W               20
           DB 00H           ;SW5,0                  21
           DB 01H           ;SW10,1                 22
           DB 02H           ;SW24,2                 23
           DB 03H           ;SW23,3                 24
          ;DB RST           ;SW24  RST KEY          25
ENTER:   CJNE   A, #0AH, ENTER1
   MOV   INPUT, A
   RET
ENTER1:   CJNE   A, #01H, ENTER2
   MOV   INPUT, A
   RET
ENTER2:   CJNE   A, #04H, ENTER3
   MOV   INPUT, A
   RET
ENTER3:   CJNE   A, #06H, ENTER4
   MOV   INPUT, A
   RET
ENTER4:   CJNE   A, #09H, NOENTER
   MOV   INPUT, A
NOENTER:   RET
```

```
MOVING:
   MOV    A, INPUT
   CJNE    A, #01H, MOVING2
   CALL    DELAY
   MOV    A, SROW1
   RLC    A
   JC    GAMEOVER
   MOV    SROW1, A
   RET
MOVING2:    CJNE    A, #04H, MOVING3
   CALL    DELAY
   MOV    A, SCOL1
   RRC    A
   JC    GAMEOVER
   MOV    SCOL1, A
   RET
MOVING3:    CJNE    A, #06H, MOVING4
   CALL    DELAY
   MOV    A, SCOL1
   RLC    A
   JC    GAMEOVER
   MOV    SCOL1, A
   RET
MOVING4:    CJNE    A, #09H, RESET
   CALL    DELAY
   MOV    A, SROW1
   RRC    A
   JC    GAMEOVER
   MOV    SROW1, A
   RET

RESET:
   JMP    INIT


GAMEOVER:
   CLR    TCON.TR0
   MOV    SROW1, #00000000B
   MOV    SCOL1, #00000000B
   CALL    DOTCOLG
```

```
LOOPG:    MOV    FOOD1, #11111111B
   MOV    FOOD2, #11111111B
   CALL    DOTCOLR
   CALL    DELAY
   MOV    FOOD1, #00000000B
   MOV    FOOD2, #00000000B
   CALL    DOTCOLR
   CALL    DELAY
LCD1:
       MOV      LROW,#01H
            MOV      LCOL,#04H
            CALL     CUR_MOV
            MOV      DPTR,#MESSAGE3
            MOV      FDPL,DPL
            MOV      FDPH,DPH
            MOV      NUMFONT,#0EH
            CALL     DISFONT

            MOV      LROW,#02H
            MOV      LCOL,#01H
            CALL     CUR_MOV
            MOV      DPTR,#MESSAGE0
            MOV      FDPL,DPL
            MOV      FDPH,DPH
            CALL     DISFONTT
            MOV      DPTR,#MESSAGEN
            MOV      FDPL,DPL
            MOV      FDPH,DPH
            MOV      NUMFONT,#15H
            CALL     DISFONT

       JMP    LOOPG


FOODING:
   MOV    A, FOOD2
   CJNE    A, SCOL1, OVER
   MOV    A, FOOD1
   CJNE    A, SROW1, OVER

   MOV A,SCORE
```

```
   MOV B,#1
   ADD A,B
   MOV SCORE,A
   DA  A
   CALL SEG

   CLR   TCON.TR0
   MOV A,TL0
   MOV B,#03H
   MUL AB
   MOV R1,A
   MOV A,#00000001B
DCOL:
   MOV R3,A
   MOV FOOD2,A
   RL A
   DJNZ R1,DCOL
   MOV A, #00000001B
DROW:
   MOV R7,A
   MOV FOOD1,A
   RL A
   DJNZ TL0,DROW
   MOV A,#00000001B
   CALL     DOTCOLR
   CALL     DOTCOLG0
   SETB   TCON.TR0

OVER:
   RET

SEG:
MOV   DPTR,#SEG3
MOVX   @DPTR,A

MOV   A,#00H
MOV   DPTR,#SEG2
MOVX   @DPTR,A

MOV   A,#00H
MOV   DPTR,#SEG1
```

```
MOVX    @DPTR,A
RET

DISFONT:     MOV       R4,#00H
FLOOP:       MOV       DPL,FDPL
             MOV       DPH,FDPH
             MOV       A,R4
             MOVC      A,@A+DPTR
             MOV       DATA,A

             CALL      DATAWR
             INC       R4
             MOV       A,R4
             CJNE      A,NUMFONT,FLOOP
       RET

DISFONTT:
   MOV    A, TIME
   MOV    B, #100
   DIV    AB
   MOV      DPL, FDPL
   MOV      DPH, FDPH
   MOVC     A, @A+DPTR
   MOV      DATA, A
   CALL     DATAWR

   MOV    A, B
   MOV    B, #10
   DIV    AB
   MOV      DPL, FDPL
   MOV      DPH, FDPH
   MOVC    A, @A+DPTR
   MOV    DATA, A
   CALL    DATAWR

   MOV    A, B
   MOV    B, #10
   DIV    AB
   MOV    A, B
   MOV      DPL, FDPL
   MOV      DPH, FDPH
```

```
   MOVC    A, @A+DPTR
   MOV    DATA, A
   CALL    DATAWR

   RET

;*************************************************************
;*        서브 루틴: 커서의 위치 제어(CUR_MOV)                *
;*             입력: 커서의 행과 열 < LROW(행) ,LCOL(열) >     *
;*             출력: LCD 화면                                 *
;*             기능: 커서 위치 조정                           *
;*************************************************************
CUR_MOV:     MOV      A,LROW
             CJNE     A,#01H, NEXT
             MOV      A ,#LINE_1
             ADD      A ,LCOL
             MOV      INST,A
             CALL     INSTWR
             JMP      RET_POINT

NEXT:        CJNE     A,#02H, RET_POINT
             MOV      A ,#LINE_2
             ADD      A ,LCOL
             MOV      INST,A
             CALL     INSTWR
RET_POINT:   RET

;*************************************************************
;*          서브 루틴: INSTWR                                *
;*               입력: INST                                  *
;*               출력: LCD 화면                              *
;*               기능: LCD INSTRUCTION 레지스터 쓰기          *
;*************************************************************
INSTWR:      CALL      INSTRD
             MOV       DPTR,#LCDWIR
             MOV       A,INST
             MOVX      @DPTR,A
             RET

;*************************************************************
;*          서브 루틴:DATAWR                                 *
```

```
;*              입력:DATA                                        *
;*              출력:LCD 화면                                    *
;*              기능:LCD DATA 레지스터 쓰기                      *
;*****************************************************************
DATAWR:     CALL      INSTRD
            MOV       DPTR,#LCDWDR
            MOV       A,DATA
            MOVX      @DPTR,A
            RET

;*****************************************************************
;*          서브 루틴:INSTRD                                     *
;*              입력:없음                                        *
;*              출력:BUSY                                        *
;*              기능:비지 플래그/어드레스 읽기                   *
;*****************************************************************
INSTRD:     MOV       DPTR,#LCDRIR
            MOVX      A,@DPTR
            JB        ACC.7,INSTRD
            RET




DOTCOLG0:
   MOV    DPTR, #COLGREEN
   MOV    A, #00
   MOVX   @DPTR, A
            MOV     DPTR,#ROW
            MOV     A, #11111111B
            MOVX    @DPTR,A
            RET

DOTCOLR0:
   MOV    DPTR, #COLRED
   MOV    A, #00H
   MOVX   @DPTR, A
            MOV     DPTR,#ROW
            MOV     A, #11111111B
            MOVX    @DPTR,A
            RET
```

```
DOTCOLG:    MOV    DPTR,#COLGREEN
            MOV    A, SCOL1
            MOVX   @DPTR,A

            MOV    DPTR,#ROW
            MOV    A,SROW1
            MOVX   @DPTR,A
            RET

DOTCOLR:    MOV    DPTR,#COLRED
            MOV    A,FOOD2
            MOVX   @DPTR,A

            MOV    DPTR,#ROW
            MOV    A,FOOD1
            MOVX   @DPTR,A
            RET

SCANN1:     PUSH   PSW     ; PSW 값을 스택에 보관

     MOV   R7, #0AH
KEYINIT1:  DJNZ   R7, FINDKEY1
     POP    PSW            ; 스택으로 부터 PSW값을 가지고 옴
     RET

FINDKEY1:  MOV    R1,#00H        ; R1의 초기화
     MOV    A,#11101111B   ; 데이터 아웃의 초기값


COLSCAN1:  MOV    R0,A           ; R0에 데이터 아웃 값 보관
     INC    R1             ; R1 열의 값 보관
     CALL   SUBKEY         ; 키 패드 입력 상태 조사
     ANL    A,#00011111B   ; 상위 3비트 제거
     XRL    A,#00011111B   ; XRL 연산000111111
     JNZ    RSCAN1         ; 누산기 값이 0 이 아니면, 행 스캔
     MOV    A,R0
     SETB   C
     RRC    A              ; 다음 열로 이동
     JNC    KEYINIT1       ; 모든 열을 스캔했으면, 다시 시작
     JMP    COLSCAN1       ; 다음 열의 스캔을 위한 분기
```

```
RSCAN1:     MOV     R2,#00H         ; R2 행의 값 보관
ROWSCAN1:   RRC     A               ; 어느 행이 "1" 로 바뀌었는지 조사
      JC      MATRIX1          ; 캐리가 발생하면, MATRIX로 분기
      INC     R2              ; 캐리가 발생하지 않으면, 다음 행으로 이동
      JMP     ROWSCAN1         ; 다음 행의 스캔을 위한 분기

MATRIX1:    MOV     A,R2            ; R2 에는 행의 값 보존
      MOV     B,#05H          ; 1행은 5열로 이루어짐
      MUL     AB              ; 2차원 배열을 1차원 배열로 값을 바꿈
      ADD     A,R1            ; R1 에는 열의 값 보존
      CALL    INDEX           ; 키 코드 값을 지정
      CALL    ENTER       ; 입력된 값을 메모리에 저장
      POP     PSW             ; 스택으로 부터 PSW값을 가지고 옴
      RET                     ; 상위 루틴으로 복귀


MESSAGE0:
   DB '0','1','2','3','4'
   DB '5','6','7','8','9'

MESSAGE1:   DB 'P','r','e','s','s'
       DB ' ','A','n','y',' '
            DB 'K','e','y',' '


MESSAGE2:   DB 'T','o',' ','S','t'
            DB 'a','r','t',' ',' '
            DB ' ',' ',' ',' ',' '
       DB ' ',' ',' '


MESSAGE3:   DB 'G','a','m','e',' '
       DB 'O','v','e','r',' '
            DB ' ',' ',' ',' '


MESSAGE4:   DB 'P','A','U','S','E'
       DB ' ',' ',' ',' ',' '
```

```
            DB ' ',' ',' ',' '


MESSAGEN:
            DB ' ',' ',' ',' ',' '
            DB ' ',' ',' ',' ',' '
            DB ' ',' ',' ',' ',' '
            DB ' ',' ',' ',' ',' '

DELAY:     MOV       R2,#002H ; 1주기
DELAY3:    MOV       R1,#0FFH ; 1주기
DELAY2:    MOV       R0,#0FFH ; 1주기
DELAY1:    DJNZ      R0,DELAY1; 2*R0 주기
           DJNZ      R1,DELAY2; (1+2*R0+2)*R1 주기
           DJNZ      R2,DELAY3; (1+(1+2*R0+2)*R1+2)*R2 주기
       RET




SERVICE0:
   JMP    PAUSE

PAUSE:
   CLR    TCON.TR0

        MOV       LROW,#01H
            MOV       LCOL,#06H
            CALL      CUR_MOV
            MOV       DPTR,#MESSAGE4
            MOV       FDPL,DPL
            MOV       FDPH,DPH
            MOV       NUMFONT,#0EH
            CALL      DISFONT

            MOV       LROW,#02H
            MOV       LCOL,#01H
            CALL      CUR_MOV
```

```
            MOV      DPTR,#MESSAGE0
            MOV      FDPL,DPL
            MOV      FDPH,DPH
            CALL     DISFONTT
            MOV      DPTR,#MESSAGEN
            MOV      FDPL,DPL
            MOV      FDPH,DPH
            MOV      NUMFONT,#15H
            CALL     DISFONT


   CALL   SCANN0
          MOV      LROW,#01H
            MOV      LCOL,#00H
            CALL     CUR_MOV
            MOV      DPTR,#MESSAGEN
            MOV      FDPL,DPL
            MOV      FDPH,DPH
            MOV      NUMFONT,#15H
            CALL     DISFONT


   CJNE    A, #00H, RESTART
   MOV     A, #01

RESTART:
   SETB    TCON.TR0
   SETB    TCON.TR1
   RETI

SERVICET0:
   CLR     TCON.TR0
   MOV     TH0, #00010000B
   MOV     TL0, #00000000B
   SETB    TCON.TR0
   DJNZ    R6, WAIT
   JMP     COUNT

WAIT:

   RETI
```

```
COUNT:
   MOV    R6, #15
   MOV    A, TIME
   INC    A
   MOV    TIME, A

            MOV      LROW,#02H
            MOV      LCOL,#01H
            CALL     CUR_MOV
            MOV      DPTR,#MESSAGE0
            MOV      FDPL,DPL
            MOV      FDPH,DPH
            CALL     DISFONTT

   RETI


ORG    9F03H
JMP    SERVICE0

ORG      9F0BH
JMP      SERVICET0

END
```