```
;
;MICROPROCESSOR:8031 INTEL
;SYSTEM:MCS-51 MACRO ASSEMBLER
;
;Stepper motor drive - hi/lo speed
;
;512 steps assumed for triangular waveform full cycle - lo speed
;8 steps assumed for hi-speed waveform full cycle
;
; Registers
;
; R0: Motor 1 step level - phase A (addr: A000)
; R1: Motor 1 step level - Phase B (addr: 9000)
; R2: Motor 2 step level - phase A (addr: E000)
; R3: Motor 2 step level - phase B (addr: D000)
; R4: Scratch pad
; R5: Scratch pad
; R6: Scratch pad
; R7: Scratch pad

DATASEG SEGMENT DATA
CODESEG SEGMENT CODE
;
;  CONSTANTS
;
DELAY:  EQU     0F8H    ; Motor fade delay
THRES:  EQU     040H    ; Motor threshold turn off
FDINC:  EQU     004H    ; Motor fade up current increment
;
; RAM addresses
;
CUR2APH: EQU    040H    ; Motor 2 - ph A current
CUR2BPH: EQU    041H    ; Motor 2 - ph B current
DIR1APH: EQU    042H    ; Motor 1 - ph A step direction (triang wave only)
DIR1BPH: EQU    043H    ; Motor 1 - ph B step direction (triang wave only)
DIR2APH: EQU    044H    ; Motor 2 - ph A step direction (triang wave only)
DIR2BPH: EQU    045H    ; Motor 2 - ph B step direction (triang wave only)
CYC1MOT: EQU    046H    ; Motor 1 hi rate cycle
CYC2MOT: EQU    047H    ; Motor 2 hi rate cycle
DIR1MOT: EQU    048H    ; Motor 1 - command direction
DIR2MOT: EQU    049H    ; Motor 2 - command direction
RATMOT:  EQU    04AH    ; Motor rates (lo/hi)
FAD2LEV: EQU    04BH    ; Motor 2 fade level
FAD1TML: EQU    04CH    ; Motor 1 fade timer low byte
FAD1TMH: EQU    04DH    ; Motor 1 fade timer high byte
FAD2TML: EQU    04EH    ; Motor 2 fade timer low byte
FAD2TMH: EQU    04FH    ; Motor 2 fade timer hi byte
FAD2CUR: EQU    050H    ; Motor 2 fade multiplier timer
TEMP:    EQU    051H    ; Temporary storage
;
MOTOR1A: EQU    0A000H  ; Address of motor 1 Phase A
MOTOR1B: EQU    09000H  ; Address of motor 1 Phase B
MOTOR2A: EQU    0E000H  ; Address of motor 2 Phase A
MOTOR2B: EQU    0D000H  ; Address of motor 2 Phase B
;
;
        RSEG    CODESEG
;
        ORG     0000H
        LJMP    INIT                    ; Reset vector
;
```

```
        ORG     0003H
        CLR     EA                      ; Disable interrupts
        AJMP    INT1MOT                 ; Motor 1 interrupt vector
;
        ORG     0013H
        CLR     EA                      ; Disable interrupts
        AJMP    INT2MOT                 ; Motor 2 interrupt vector
;
        ORG     0100H
HIWAV   EQU     $
        DB      00FFH                   ; Hi-speed drive waveform
        DB      00FFH
        DB      00FFH
        DB      0080H
        DB      0000H
        DB      0000H
        DB      0000H
        DB      0080H
;
        ORG     0200H
INIT:   CLR     EA
        CLR     EX0                     ; Disable interrupt
        CLR     EX1                     ;
        CLR     ET0                     ;
        CLR     ET1                     ;
        CLR     ES                      ;
        SETB    IT0                     ; Allow int0 edge triggered
        SETB    IT1                     ; Allow int1 edge triggered
        SETB    PX0                     ; Set high priority for int 0
        SETB    PX1                     ; Set high priority for int 1
;
        MOV     P1, #0FFH               ; Set port 1 for input
        MOV     R0, #040H               ; Set 90 degree phase difference
        MOV     R1, #0C0H               ; on motor 1.
        MOV     R2, #040H               ; Set 90 degree phase difference
        MOV     R3, #0C0H               ; on motor 2.
        MOV     CUR2APH, R2
        MOV     CUR2BPH, R3
        MOV     DIR1APH, #00            ; Set direction of triangle wave all phases a
nd motors
        MOV     DIR1BPH, #00
        MOV     DIR2APH, #00
        MOV     DIR2BPH, #00
        MOV     CYC1MOT, #00            ; Set hi rate motor 1 cycle
        MOV     CYC2MOT, #00            ; Set hi rate motor 2 cycle
        MOV     DIR1MOT, #00            ; Motor 1 command direction
        MOV     DIR2MOT, #00            ; Motor 2 command direction
        MOV     FAD2LEV, #00            ; Motor 2 fade level
        MOV     FAD1TML, #00
        MOV     FAD1TMH, #0FFH
        MOV     FAD2TML, #00
        MOV     FAD2TMH, #DELAY
        MOV     FAD2CUR, #00
;
        SETB    EX0                     ; Enable int0
        SETB    EX1                     ; Enable int1
        SETB    EA                      ; Enable interrupts
;
;
;  NON INTERRUPT ROUTINE - MAIN PROGRAM
;
```

```
;
;  DETERMINE HOST COMMANDS
;
MAIN:   JB      P1.2,SETRAT             ; Jump if bit set
        MOV     RATMOT, #00H            ; Clear motor rate
        SJMP    CHKLM1                  ; Jump to limit check
SETRAT: MOV     RATMOT, #01H            ; Set motor rate
;
BITMT1: JB      P1.0,SETDR1             ; Jump if bit set
        MOV     DIR1MOT, #00H           ; Clear motor 1 direction
        SJMP    BITMT2
SETDR1: MOV     DIR1MOT, #01H           ; Set motor 1 direction
;
BITMT2: JB      P1.1,SETDR2             ; Jump if bit set
        MOV     DIR2MOT, #00H           ; Clear motor 2 direction
        SJMP    FADE1
SETDR2: MOV     DIR2MOT, #01H           ; Set motor 2 direction
        SJMP    FADE1
;
CHKLM1: CJNE    R0, #000H, C11          ; If phase A or B motor 1 at
        SJMP    CHKLM2                  ; limits, bypass direction
C11:    CJNE    R0, #0FFH, C12          ; limits, bypass direction
        SJMP    CHKLM2
C12:    CJNE    R1, #000H, C13
        SJMP    CHKLM2
C13:    CJNE    R1, #0FFH, C14
        SJMP    CHKLM2
C14:    JB      P1.0,SETDRA             ; Jump if bit set
        MOV     DIR1MOT, #00H           ; Clear motor 1 direction
        SJMP    CHKLM2
SETDRA: MOV     DIR1MOT, #01H           ; Set motor 1 direction
;
CHKLM2: CJNE    R2, #000H, C21          ; If phase A or B motor 1 at
        SJMP    FADE1                   ; limits, bypass direction
C21:    CJNE    R2, #0FFH, C22          ; limits, bypass direction
        SJMP    FADE1
C22:    CJNE    R3, #000H, C23
        SJMP    FADE1
C23:    CJNE    R3, #0FFH, C24
        SJMP    FADE1
C24:    JB      P1.1,SETDRB             ; Jump if bit set
        MOV     DIR2MOT, #00H           ; Clear motor 2 direction
        SJMP    FADE1
SETDRB: MOV     DIR2MOT, #01H           ; Set motor 2 direction
;
FADE1:  MOV     A, FAD1TMH              ; Load hi byte fade timer
        JZ      FADE11                  ; If zero, bypass decrement
        DJNZ    FAD1TML, FADE2          ; Decr low byte, jump if not 0
        DJNZ    FAD1TMH, FADE2          ; Decrement hi byte, if not 0 bypass
FADE11: CLR     EX0                     ; Prevent interrupts
        MOV     A, #80H                 ; Clear accumulator
        MOV     DPTR, #MOTOR1A          ; Zero motor current both phase
        MOVX    @DPTR, A
        MOV     A, #80H                 ; Clear accumulator
        MOV     DPTR, #MOTOR1B
        MOVX    @DPTR, A
        SETB    EX0                     ; Enable interrupts
        CLR     EX1                     ; Prevent interrupts
        MOV     A, #80H                 ; Clear accumulator
        MOV     DPTR, #MOTOR2A          ; Zero motor current both phase
        MOVX    @DPTR, A
```

```
        MOV     A, #80H                 ; Clear accumulator
        MOV     DPTR, #MOTOR2B
        MOVX    @DPTR, A
        SETB    EX1                     ; Enable interrupts
        AJMP    MAIN
;
FADE2:  MOV     A, FAD2TMH              ; Load hi byte fade timer
        JZ      DECFD2                  ; If hi byte zero do not decrement
        DJNZ    FAD2TML, MN             ; Decrement motor 2 time, lo byte zero?
        DJNZ    FAD2TMH, MN             ; Otherwise, Decrement hi byte
DECFD2: DJNZ    FAD2CUR, MN             ; Fade current multiplier timer, zero?
        MOV     A, FAD2LEV              ; Otherwise, dec level if above threshold
        CJNE    A, #THRES, DECFD21
MN:     AJMP    MAIN
DECFD21:
        DEC     FAD2LEV
        MOV     A, FAD2LEV              ; Adjust offset from fade gain
        CPL     A                       ; Complement level
        CLR     C                       ; Clear carry for division
        RRC     A                       ; Divide by two offset
        INC     A                       ; Shape to 80H
        MOV     R7, A                   ; Store offset result
;
        MOV     A, CUR2APH
        MOV     B, FAD2LEV
        MUL     AB                      ; Multiply phase A and level fade
        MOV     A, B                    ; Move high order of result to acc
        ADD     A, R7                   ; Add offset to result
        MOV     R5, A                   ; Store result
;
        MOV     A, CUR2BPH
        MOV     B, FAD2LEV
        MUL     AB                      ; Multiply phase B and level fade
        MOV     A, B                    ; Move high order of result to acc
        ADD     A, R7                   ; Add offset to result
        MOV     R6, A                   ; Store result
;
        MOV     A, FAD2TMH              ; Check interrupt timer override
        CJNE    A, #0FFH, FAD2MOT       ; If reset, do not fade motor
        AJMP    MAIN
FAD2MOT:
        CLR     EX1
        MOV     A, R5                   ; Set Motor 2 phase A
        MOV     DPTR, #MOTOR2A
        MOVX    @DPTR, A
        MOV     A, R6                   ; Set motor 2 phase B
        MOV     DPTR, #MOTOR2B
        MOVX    @DPTR, A
        SETB    EX1
;
FADOUT: AJMP    MAIN                    ; Continue cycle
;
;
;
;  MOTOR 1 CONTROL INTERRUPT ROUTINE
;
INT1MOT:
        CLR     EX0
        PUSH    ACC                     ; Save registers
        PUSH    B
        PUSH    PSW
```

```
        PUSH    DPL
        PUSH    DPH
;
        MOV     FAD1TMH, #0FFH          ; Set timer high byte
        MOV     A, RATMOT               ; Get motor rate (hi/lo)
        JZ      LOMOT1                  ; Jump if set lo
;
;  HI RATE DRIVE
;
HIMOT1: MOV     DPTR, #HIWAV            ; Set data pointer to wave table
        MOV     A, CYC1MOT              ; Get motor 1 cycle
        ANL     A, #07H                 ; Reduce to 8 step cycle
        MOV     R4, A                   ; Save accumulator
        MOVC    A, @A+DPTR              ; Get phase A from table
        MOV     TEMP, A                 ; Store phase A current
        MOV     A, R4                   ; Retrieve accumulator
        CLR     C                       ; Clear carry
        ADD     A, #02H                 ; Add 90 degree phase shift
        ANL     A, #07H                 ; Reduce to 8 step cycle
        MOVC    A, @A+DPTR              ; Get phase B from table
;
        MOV     DPTR, #MOTOR1B          ; Set Motor 1 phase B
        MOVX    @DPTR, A
        MOV     A, TEMP                 ; Set motor 1 phase A
        MOV     DPTR, #MOTOR1A
        MOVX    @DPTR, A
;
        MOV     A, DIR1MOT              ; Get motor direction
        JNZ     HIDIR1                  ; Jump if zero
        INC     CYC1MOT                 ; Increment motor cycle
        SJMP    M1OUT2
HIDIR1: DEC     CYC1MOT                 ; Decrement motor cycle
        SJMP    M1OUT2
;
;  LO RATE DRIVE - TRIANGULAR WAVE
;
;  PHASE A MOTOR 1
;
LOMOT1: MOV     A, DIR1APH              ; Direction of phase A.
        XRL     A, DIR1MOT              ; Determine count direction,
        JZ      M1ADEC                  ; Jump to decrement if zero
        INC     R0                      ; Increment phase A.
        CJNE    R0, #0FFH, M1BPH        ; Check if phase A at FFH,
        SJMP    M1ATOG                  ; Toggle direction if so.
M1ADEC: DEC     R0                      ; Decrement phase A
        CJNE    R0, #000H, M1BPH        ; Check if phase A at FFH
M1ATOG: XRL     DIR1APH, #01H           ; If so, toggle phase direction
;
;  PHASE B MOTOR 1
;
M1BPH:  MOV     A, DIR1BPH              ; Direction of phase B.
        XRL     A, DIR1MOT              ; Determine count direction,
        JZ      M1BDEC                  ; Jump to decrement if zero
        INC     R1                      ; Increment phase B.
        CJNE    R1, #0FFH, M1OUT        ; Check if phase B at FFH,
        SJMP    M1BTOG                  ; Toggle direction if so.
M1BDEC: DEC     R1                      ; Decrement phase B
        CJNE    R1, #000H, M1OUT        ; Check if phase B at FFH
M1BTOG: XRL     DIR1BPH, #01H           ; If so, toggle phase direction
;
M1OUT:  MOV     A, R0                   ; Set Motor 1 phase A
```

```
        MOV     DPTR, #MOTOR1A
        MOVX    @DPTR, A
        MOV     A, R1                   ; Set motor 1 phase B
        MOV     DPTR, #MOTOR1B
        MOVX    @DPTR, A
;
M1OUT2: POP     DPH                     ; retrieve registers
        POP     DPL
        POP     PSW
        POP     B
        POP     ACC
;
        SETB    EX0
        SETB    EA                      ; Enable interrupts
        RETI                            ; Exit interrupt
;
;
;  MOTOR 2 INTERRUPT ROUTINE
;
INT2MOT:
        CLR     EX1
        PUSH    ACC                     ; Save registers
        PUSH    B
        PUSH    PSW
        PUSH    DPL
        PUSH    DPH
;
        MOV     FAD2TMH, #DELAY         ; Apply full motor control
        MOV     A, RATMOT               ; Get motor rate (hi/lo)
        JZ      LOMOT2                  ; Jump if set lo
;
HIMOT2: MOV     DPTR, #HIWAV            ; Set data pointer to wave table
        MOV     A, CYC2MOT              ; Get motor 2 cycle
        ANL     A, #07H                 ; Reduce to 8 step cycle
        MOV     R4, A                   ; Save accumulator
        MOVC    A, @A+DPTR              ; Get phase A from table
        MOV     CUR2APH, A              ; Store phase A current
        MOV     A, R4                   ; Retrieve accumulator
        CLR     C                       ; Clear carry
        ADD     A, #02H                 ; Add 90 degree phase shift
        ANL     A, #07H                 ; Reduce to 8 step cycle
        MOVC    A, @A+DPTR              ; Get phase B from table
        MOV     CUR2BPH, A              ; Store phase B current
;
        MOV     A, DIR2MOT              ; Get motor direction
        JNZ     HIDIR2                  ; Jump if zero
        INC     CYC2MOT                 ; Increment motor cycle
        SJMP    M2OUT
HIDIR2: DEC     CYC2MOT                 ; Decrement motor cycle
        SJMP    M2OUT
;
;  LO RATE DRIVE - TRIANGULAR WAVE
;
;  PHASE A MOTOR 2
;
LOMOT2: MOV     A, DIR2APH              ; Direction of phase A.
        XRL     A, DIR2MOT              ; Determine count direction,
        JZ      M2ADEC                  ; Jump to decrement if zero
        INC     R2                      ; Increment phase A.
        CJNE    R2, #0FFH, M2ASAV       ; Check if phase A at FFH,
        SJMP    M2ATOG                  ; Toggle direction if so.
```

```
M2ADEC: DEC     R2                      ; Decrement phase A
        CJNE    R2, #000H, M2ASAV       ; Check if phase A at FFH
M2ATOG: XRL     DIR2APH, #01H           ; If so, toggle phase direction
M2ASAV: MOV     CUR2APH, R2             ; Store ph A current
;
;  PHASE B MOTOR 2
;
        MOV     A, DIR2BPH              ; Direction of phase B.
        XRL     A, DIR2MOT              ; Determine count direction,
        JZ      M2BDEC                  ; Jump to decrement if zero
        INC     R3                      ; Increment phase B.
        CJNE    R3, #0FFH, M2BSAV       ; Check if phase B at FFH,
        SJMP    M2BTOG                  ; Toggle direction if so.
M2BDEC: DEC     R3                      ; Decrement phase B
        CJNE    R3, #000H, M2BSAV       ; Check if phase B at FFH
M2BTOG: XRL     DIR2BPH, #01H           ; If so, toggle phase direction
M2BSAV: MOV     CUR2BPH, R3             ; Store ph B current
;
        MOV     A, FAD2LEV              ; Adjust offset from fade gain
        ADD     A, #FDINC               ; Fade up motor fade level
        JNC     RAMP21                  ; If no overflow, bypass
        MOV     FAD2LEV, #0FFH          ; If overflow, set to FFH
        SJMP    M2OUT                   ; Bypass fade multiplier
RAMP21: MOV     FAD2LEV, A              ; Save increment
        CPL     A                       ; Complement level
        CLR     C                       ; Clear carry for division
        RRC     A                       ; Divide by two offset
        INC     A                       ; Shape to 80
        MOV     R4, A                   ; Store offset result
;
        MOV     A, R2
        MOV     B, FAD2LEV
        MUL     AB                      ; Multiply phase A and level fade
        MOV     A, B                    ; Move high order of result to acc
        ADD     A, R4                   ; Add offset to result
        MOV     CUR2APH, A              ; Store result
;
        MOV     A, R3
        MOV     B, FAD2LEV
        MUL     AB                      ; Multiply phase B and level fade
        MOV     A, B                    ; Move high order of result to acc
        ADD     A, R4                   ; Add offset to result
        MOV     CUR2BPH, A              ; Store result
;
M2OUT:  MOV     A, CUR2APH              ; Set Motor 2 phase A
        MOV     DPTR, #MOTOR2A
        MOVX    @DPTR, A
        MOV     A, CUR2BPH              ; Set motor 2 phase B
        MOV     DPTR, #MOTOR2B
        MOVX    @DPTR, A
;
        POP     DPH
        POP     DPL
        POP     PSW
        POP     B
        POP     ACC
;
        SETB    EX1
        SETB    EA                      ; Enable interrupts
        RETI                            ; Exit interrupt
;
```

```
;
        END
;
```