```
;*******************************************************************************
;* Lab 5 Main [includes LibV2.2]                                               *
;*******************************************************************************
;* Summary: DC Motor PI Controller                                             *
;*   -                                                                         *
;*                                                                             *
;* Author: BENJAMIN FIELDS, MILES ALDERMAN                                     *
;*   Cal Poly University                                                       *
;*   Spring 2022                                                               *
;*                                                                             *
;* Revision History:                                                           *
;*   -                                                                         *
;*                                                                             *
;* ToDo:                                                                       *
;*   -                                                                         *
;*******************************************************************************

;/-----------------------------------------------------------------------------\
;| Include all associated files                                                |
;\-----------------------------------------------------------------------------/
; The following are external files to be included during assembly
              XDEF  main
              XDEF  Theta_OLD, RUN, CL, V_ref, KP, KI, UPDATE_FLG1
;/-----------------------------------------------------------------------------\
;| External References                                                         |
;\-----------------------------------------------------------------------------/
; All labels from other files must have an external reference

              XREF  ENABLE_MOTOR, DISABLE_MOTOR
              XREF  STARTUP_MOTOR, UPDATE_MOTOR, CURRENT_MOTOR
              XREF  STARTUP_PWM, STARTUP_ATD0, STARTUP_ATD1
              XREF  OUTDACA, OUTDACB
              XREF  STARTUP_ENCODER, READ_ENCODER
              XREF  DELAY_MILLI, DELAY_MICRO
              XREF  INITLCD, SETADDR, GETADDR, CURSOR_ON, DISP_OFF
              XREF  OUTCHAR, OUTCHAR_AT, OUTSTRING, OUTSTRING_AT
              XREF  INITKEY, LKEY_FLG, GETCHAR
              XREF  LCDTEMPLATE, UPDATELCD_L1, UPDATELCD_L2
              XREF  LVREF_BUF, LVACT_BUF, LERR_BUF,LEFF_BUF, LKP_BUF, LKI_BUF
              XREF  Entry, ISR_KEYPAD

              XREF  V_act_DISP, ERR_DISP, EFF_DISP
              XREF  FREDENTRY

;/-----------------------------------------------------------------------------\
;| Assembler Equates                                                           |
;\-----------------------------------------------------------------------------/
; Constant values can be equated here
```

```
TFLG1         EQU   $004E
TC0           EQU   $0050
C0F           EQU   %00000001          ; timer channel 0 output compare bit
PORTT         EQU   $0240              ; PORTT pin 8 to be used for interrupt timing
LOWER_LIM     EQU   -625               ; number for max reverse duty cycle
UPPER_LIM     EQU   625                ; number for max forward duty cycle
INTERVAL      EQU   $4E20              ; number of clock pulses that equal 2ms from
10.2MHz clock


;/-------------------------------------------------------------------------------\
;| Variables in RAM                                                              |
;\-------------------------------------------------------------------------------/
; The following variables are located in unpaged ram

DEFAULT_RAM:  SECTION

RUN:          DS.B  1                  ; Boolean indicating controller is running
CL:           DS.B  1                  ; Boolean for closed-loop active

V_ref:        DS.W  1                  ; reference velocity
V_act:        DS.W  1                  ; actual velocity
Theta_OLD:    DS.W  1                  ; previous encoder reading
Theta_NEW:    DS.W  1                  ; current encoder reading
ERR:          DS.W  1                  ; current error
EFF:          DS.W  1                  ; current effort
KP:           DS.W  1                  ; proportional gain
KI:           DS.W  1                  ; integral gain
e:            DS.W  1                  ; voltage feedback error
e_sum:        DS.W  1                  ; sum of voltage feedback error in current run
e_p:          DS.W  1                  ; proportional error for SDBA
e_i:          DS.W  1                  ; integral error for SDBA
a_out:        DS.W  1                  ; controller output
a_star:       DS.W  1                  ; bounded PWM output
TEMP:         DS.W  1
UPDATE_FLG1   DS.B  1                  ; Boolean for display update for line one
UPDATE_COUNT  DS.B  1                  ; counter for display update timing



;/-------------------------------------------------------------------------------\
;|  Main Program Code                                                            |
;\-------------------------------------------------------------------------------/
; Your code goes here

MyCode:       SECTION
main:
        clrw   e
        clrw   e_sum
        clrw   e_p
```

```
        clrw   e_i
        clrw   a_out
        clrw   a_star
        clrw   V_ref
        clrw   V_act
        clrw   Theta_OLD
        clrw   Theta_NEW
        clrw   KP
        clrw   KI
        clrw   ERR
        clrw   EFF
        clr    UPDATE_FLG1
        clr    UPDATE_COUNT
        clr    RUN
        clr    CL
        bgnd
        jsr   FREDENTRY
spin:   bra   spin                    ; endless horizontal loop

TC0ISR:
        bset  PORTT, $80              ; turn on PORTT pin 8 to begin ISR timing

        inc   UPDATE_COUNT            ; unless UPDATE_COUNT = 0, skip saving
        bne   measurements            ;   display variables
        movw  V_act, V_act_DISP       ; take a snapshot of variables to enable
        movw  ERR, ERR_DISP           ;   consistent display
        movw  EFF, EFF_DISP
        movb  #$01, UPDATE_FLG1       ; set UPDATE_FLG1 when appropriate

; Measurements block

measurements:
; Read encoder value
        jsr   READ_ENCODER            ; read encoder position
        std   Theta_NEW               ; store it

; Compute 2-point difference to get speed
        subd  Theta_OLD               ; compute displacement since last reading
        std   V_act                   ; store displacement as actual speed
        movw  Theta_NEW, Theta_OLD    ; move current reading to previous reading

error_measurement:
        ldd   V_ref
        subd  V_act
        std   e                       ; find current error between V_ref and V_act
        std   ERR

open_closed_loop:
        tst   CL
        bne   proportional_control
```

```
        ldd   V_ref
        std   e
        subd  V_act
        std   ERR

proportional_control:
        ldd   e
        ldy   KP
        emuls                       ; multiply error by proportional gain * 1024
        ldx   #$0400
        edivs                       ; divide by 1024 for true magnitude
        sty   e_p                   ; store in proportional error

integral_control:
        ldd   e
        ldy   e_sum                 ; add current error to previous error
        jsr   SDBA                  ; SDBA
        std   e_sum
        ldy   KI
        emuls                       ; multiply error sum by integral gain * 1024
        ldx   #$0400
        edivs                       ; divide by 1024 for true magnitude
        sty   e_i                   ; store in integral error

controller_combine:
        ldy   e_i
        ldd   e_p
        jsr   SDBA                  ; add proportional and integral error for control
output
        std   a_out

PWM_saturation:                     ; translate control output to PWM resolution at
saturation
        ldx   a_out
        tstx
        bmi   PWM_N
        cpx   #$0271
        ble   PWM_exit
        ldx   #$0271
        bra   PWM_exit

PWM_N:
        ldy   a_out
        negy
        cpy   #$0271
        ble   PWM_exit
        ldx   #$FD8F

PWM_exit:
        stx   a_star
```

```
EFF_determine:
        ldd   a_star
        ldy   #$0064
        emuls
        ldx   #$0271
        edivs
        sty   EFF

RUN_DETERMINE:
        tst   RUN
        beq   MOTOR_OFF
        ldd   a_star
        bra   MOTOR_OUT

MOTOR_OFF:
        ldd   #$0000
        movw  #$0000, e_sum
        movw  #$0000, e

MOTOR_OUT:
        jsr   UPDATE_MOTOR
        ldd   TC0                     ; load $0044:$0045 into d
        addd  #INTERVAL                ; add interval to timer count
        std   TC0                      ; store in timer channel 0
        bset  TFLG1, C0F               ; clear timer output compare flag by writing a 1
to it

Reponse_measure:
        ldd   V_act
        ldy   #13
        emuls
        addd  #2048
        jsr   OUTDACA

        rti




;/---------------------------------------------------------------------------\
;| Subroutines                                                               |
;\---------------------------------------------------------------------------/
; General purpose subroutines go here

SDBA:

        pshx
        pshc
```

```
addroutine:
        sty   TEMP
        addd  TEMP                      ;add two operands
        bvs   overflowdesignate         ;if 2's complement overflow, saturated

return:
        pulc
        pulx
        rts

overflowdesignate:
        cpy   #$0000
        bmi   neg_OF                    ;if N flag true (negative), negative overflow
        ldd   #$7FFF                    ;else, positive overflow
        bra   return

neg_OF:
        ldd   #$8000
        bra   return


;/-------------------------------------------------------------------------------\
;| ASCII Messages and Constant Data                                              |
;\-------------------------------------------------------------------------------/
; Any constants can be defined here


;/-------------------------------------------------------------------------------\
;| Vectors                                                                       |
;\-------------------------------------------------------------------------------/
; Add interrupt and reset vectors here
         ORG  $FFFE                     ; reset vector address
         DC.W Entry
         ORG  $FFEE
         DC.W TC0ISR
```