list p=16f877 ; list directive to define processor

#include <p16f877.inc> ; processor specific variable definitions

\_\_CONFIG \_CP\_OFF & \_WDT\_OFF & \_BODEN\_ON & \_PWRTE\_ON & \_HS\_OSC & \_WRT\_ENABLE\_OFF & \_LVP\_ON &

\_DEBUG\_OFF & \_CPD\_OFF

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* Define variable storage

;\*

CBLOCK 0x20

ADC ; PWM threshold is ADC result

LastSensor ; last read motor sensor data

DriveWord ; six bit motor drive data

ENDC

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* Define I/O

;\*

#define OffMask B'11010101'

#define DrivePort PORTC

#define DrivePortTris TRISC

#define SensorMask B'00000111'

#define SensorPort PORTE

#define DirectionBit PORTA,1

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

org 0x000 ; startup vector

nop ; required for ICD operation

clrf PCLATH ; ensure page bits are cleared

goto Initialize ; go to beginning of program

ORG 0x004 ; interrupt vector location

retfie ; return from interrupt

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* Initialize I/O ports and peripherals

;\*

Initialize

clrf DrivePort ; all drivers off

banksel TRISA

; setup I/O

clrf DrivePortTris ; set motor drivers as outputs

movlw B'00000011' ; A/D on RA0, Direction on RA1, Motor sensors on RE<2:0>

movwf TRISA ;

; setup Timer0

movlw B'11010000' ; Timer0: Fosc, 1:2

movwf OPTION\_REG

; Setup ADC (bank1)

movlw B'00001110' ; ADC left justified, AN0 only

movwf ADCON1

banksel ADCON0

; setup ADC (bank0)

movlw B'11000001' ; ADC clock from int RC, AN0, ADC on

movwf ADCON0

bsf ADCON0,GO ; start ADC

clrf LastSensor ; initialize last sensor reading

call Commutate ; determine present motor position

clrf ADC ; start speed control threshold at zero until first ADC

reading

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* Main control loop

;\*

Loop

call ReadADC ; get the speed control from the ADC

incfsz ADC,w ; if ADC is 0xFF we're at full speed - skip timer add

goto PWM ; add Timer0 to ADC for PWM

movf DriveWord,w ; force on condition

goto Drive ; continue PWM

movf ADC,w ; restore ADC reading

addwf TMR0,w ; add it to current Timer0

movf DriveWord,w ; restore commutation drive data

btfss STATUS,C ; test if ADC + Timer0 resulted in carry

andlw OffMask ; no carry - suppress high drivers

Drive

movwf DrivePort ; enable motor drivers

call Commutate ; test for commutation change

goto Loop ; repeat loop

ReadADC

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* If the ADC is ready then read the speed control potentiometer

;\* and start the next reading

;\*

btfsc ADCON0,NOT\_DONE ; is ADC ready?

return ; no - return

movf ADRESH,w ; get ADC result

bsf ADCON0,GO ; restart ADC

movwf ADC ; save result in speed control threshold

return ;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* Read the sensor inputs and if a change is sensed then get the

;\* corresponding drive word from the drive table

;\*

Commutate

movlw SensorMask ; retain only the sensor bits

andwf SensorPort,w ; get sensor data

xorwf LastSensor,w ; test if motion sensed

btfsc STATUS,Z ; zero if no change

return ; no change - back to the PWM loop

xorwf LastSensor,f ; replace last sensor data with current

btfss DirectionBit ; test direction bit

goto FwdCom ; bit is zero - do forward commutation

; reverse commutation

movlw HIGH RevTable ; get MS byte of table

movwf PCLATH ; prepare for computed GOTO

movlw LOW RevTable ; get LS byte of table

goto Com2

FwdCom ; forward commutation

movlw HIGH FwdTable ; get MS byte of table

movwf PCLATH ; prepare for computed GOTO

movlw LOW FwdTable ; get LS byte of table

Com2

addwf LastSensor,w ; add sensor offset

btfsc STATUS,C ; page change in table?

incf PCLATH,f ; yes - adjust MS byte

call GetDrive ; get drive word from table

movwf DriveWord ; save as current drive word

return

GetDrive

movwf PCL

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;\*

;\* The drive tables are built based on the following assumptions:

;\* 1) There are six drivers in three pairs of two

;\* 2) Each driver pair consists of a high side (+V to motor) and low side (motor to ground) drive

;\* 3) A 1 in the drive word will turn the corresponding driver on

;\* 4) The three driver pairs correspond to the three motor windings: A, B and C

;\* 5) Winding A is driven by bits <1> and <0> where <1> is A's high side drive

;\* 6) Winding B is driven by bits <3> and <2> where <3> is B's high side drive

;\* 7) Winding C is driven by bits <5> and <4> where <5> is C's high side drive

;\* 8) Three sensor bits constitute the address offset to the drive table

;\* 9) A sensor bit transitions from a 0 to 1 at the moment that the corresponding

;\* winding's high side forward drive begins.

;\* 10) Sensor bit <0> corresponds to winding A

;\* 11) Sensor bit <1> corresponds to winding B

;\* 12) Sensor bit <2> corresponds to winding C

;\*

FwdTable

retlw B'00000000' ; invalid

retlw B'00010010' ; phase 6

retlw B'00001001' ; phase 4

retlw B'00011000' ; phase 5

retlw B'00100100' ; phase 2

retlw B'00000110' ; phase 1

retlw B'00100001' ; phase 3

retlw B'00000000' ; invalid

RevTable

retlw B'00000000' ; invalid

retlw B'00100001' ; phase /6

retlw B'00000110' ; phase /4

retlw B'00100100' ; phase /5

retlw B'00011000' ; phase /2

retlw B'00001001' ; phase /1

retlw B'00010010' ; phase /3

retlw B'00000000' ; invalid

END ; directive 'end of program'