```
;*****************************************************************
;PROGRAM                 :       V/F control of 1-Phase Induction Motor
;MICROCONTROLLER         :       PIC18F452
;CRYSTAL FREQUENCY       :       20MHz
;DRIVER IC USED          :       IR21362
;MOSFET used             :       6N60
;*****************************************************************
;ORIGINAL AUTHOR         :       Padmaraja Yedamale , Appliance Group
;MODIFICATIONS           :       Marco Salvatori Artezan
;ORIGINAL DATE           :       20-Dec-2001
;ACTUAL DATE             :       30-ABR-2003
;OLD Version             :       V1.0
;NEW VERSION             :       V3.2
;*****************************************************************
;Description:-
;------------
;By varying the frequency of supply to the Induction motor we can control the
;speed of the motor. But by varying only frequency, changes the flux
;also which results in changes in torque. So to keep the flux, hence
;torque, the magnitude of voltage applied also needs to be changed. Or in other
;words, the ration of V/F needs to be kept constant throughout the
;operating range.
;To control a 1 phase Induction motor, it needs 1 PWM1 to control the 1
;phase inverter plus a semi cycle control. PIC18F452 has 2 hardware PWMs.
;Semi cycle control was done by software and 2 outputs to two of the port pins.
;
;-----------------------------------------------------------------
        include         <p18f452.inc>
        include         <3im_vf.inc>
;-----------------------------------------------------------------

#define TIMER0_OV_FLAG          0
#define OFFSET2_FLAG            5
#define MOTOR_RUNNING           1
#define DEBOUNCE                0
#define ON_OFF_FLAG             1

;=================================================================
;RAM locations in Access bank, uninitialized
        UDATA_ACS


TABLE_OFFSET2   res     1       ;Phase2 offset to the Sine table(0)
FLAGS           res     1       ;Flags registers used to indicate different status
FLAGS1          res     1
FREQ_REF_H      res     1       ;Referance Frequency input in counts
FREQ_REF_L      res     1
```

```
FREQUENCY       res     1       ;Registro donde se acumula la frecuencia
CONTADOR1       res     1
FRECUENCIA      res     1
TEMPORAL        res     1
TEMPORAL1       res     1
TEMPORAL2       res     1
TEMPORAL3       res     1
ACUMULADOR      res     1
SUMADOR         res     1


CCPR2L_TEMP     res     1
TEMP            res     1
TEMP1           res     1
TEMP_LOCATION   res     2
SINE_TABLE      res     0x14    ;Sine table
;-----------------------------------------------------------------
STARTUP     code 0x00
        goto    Start                   ;Reset Vector address

        CODE 0x08
        goto    ISR_HIGH                ;Higher priority ISR at 0x0008

PRG_LOW     CODE 0x018
        goto    ISR_LOW                 ;Lower priority ISR at 0x0018

;*****************************************************************
PROG1       code
Start
;*****************************************************************
;Initialization of the Ports and timers
        clrf    FLAGS                   ;Clear the flags
        clrf    TRISC                   ;PORTC all output
        clrf    PORTC
        clrf    TRISD                   ;PORTD all output
        clrf    PORTD
        clrf    TRISE                   ;PORTE all output
        clrf    PORTE
        clrf    TRISB                   ;PORTB all output
        clrf    PORTB
        bsf     TRISB,FAULT_BIT         ;Fault input from driver
        movlw   0x0C
        movwf   CCP2CON                 ;CCP2 configured to PWM
        bsf     T2CON,2                 ;Timer2 ON
        movlw   PR2_VALUE               ;Load PR2 value to PR2 register
        movwf   PR2
        movlw   0x90
```

```
        movwf   CCPR2L
        movlw   0x24
        movwf   CONTADOR1
        bcf     PORTE,0
        bcf     PORTE,1

        call    Init_Motor_Parameters       ;Initialize motor parameters
        call    COPY_TABLE_TO_RAM
;*****************************************************************
;Initialize ADC registers
        movlw   ADCON0_VALUE                ;From ".inc" file
        movwf   ADCON0
        movlw   ADCON1_VALUE                ;From ".inc" file
        movwf   ADCON1
        movlw   0x33                        ;RA0,RA4,RA5 inputs,RA2&RA3-Outputs
        movwf   TRISA;

        bsf     DRIVER_ENABLE_PORT,DRIVER_ENABLE_BIT
        ;Enable the driver chip

;*****************************************************************
;Timre0 Initialization with prescaler
;*****************************************************************
        movlw   0X83                        ;Load the T0CON with value
        movwf   T0CON                       ;TMR0 ON and prescaler is 1:16

        movlw   0xF8                        ;Timer0 Initialization
        movwf   TMR0H
        movlw   0x5E
        movwf   TMR0L


;-------------------------------------
        bsf     INTCON,TMR0IE               ;Timer0 overflow Interrupt enable
        bsf     PIE1,TMR2IE                 ;"Timer2 to PR2 match" Interrupt enable

        bsf     PIE1,ADIE                   ;AD Converter over Interrupt enable
        bcf     IPR1,ADIP                   ;Low priority for ADC interrupt
        bsf     INTCON,RBIE                 ;PortB interrupt enable with low priority
        bcf     INTCON2,RBIP                ;for Fault checking
        movlw   0x093                       ;Power ON reset status bit/Brownout reset
        movwf   RCON                        ;status bit and Instruction flag bits are set
                                            ;Priority level on Interrupots enabled


        bsf     INTCON,PEIE                 ;Port interrupts enable
        bsf     INTCON,GIE                  ;Global interrupt enable
```

```
;*****************************************************************
;Main loop where the program will be looping
MAIN_LOOP
        btfss   FLAGS,TIMER0_OV_FLAG        ;back from Timer0 overflow?
        bra     bypass                      ;No
        call    UPDATE_PWM_DUTYCYCLES       ;Yes, update the PWM duty cycle
                                            ;with new value
        call    UPDATE_TABLE_OFFSET         ;Update 3 offsets
        bcf     FLAGS,TIMER0_OV_FLAG        ;Clear the flag
bypass
        call    SET_ADC_GO                  ;Start AD conversion
        call    KEY_CHECK                   ;Check keys change
        call    DESPLIEGA
        bra     MAIN_LOOP
;*****************************************************************
;Higher priority interrupt service routine
;"Timer2 to PR2 match", "Timer1 overflow" and Timer0 overflow are checked
;*****************************************************************
ISR_HIGH
        btfsc   PIR1,TMR2IF                 ;Timer2 to PR2 match?
        bra     TIMER2_PR2_Match
        btfsc   INTCON,TMR0IF               ;Timer0 overflow Interrupt?
        bra     TIMER0_OVERFLOW             ;Yes
        RETFIE      FAST

;*****************************************************************
TIMER2_PR2_Match
        bcf     PIR1,TMR2IF
        RETFIE      FAST

;*****************************************************************
TIMER0_OVERFLOW                             ;TMR0 overflow ISR
        movff   FREQ_REF_H,TMR0H            ;Load the Higher byte of
                                            ;SpeedCommand to TMR0H
        movff   FREQ_REF_L,TMR0L            ;Load the Lower byte of
                                            ;SpeedCommand to TMR0L

        bsf     FLAGS,TIMER0_OV_FLAG
        bcf     INTCON,TMR0IF               ;Clear TMR0IF

        decfsz  CONTADOR1,1                 ;el contador se inicializo en 18 (hz)
        RETFIE      FAST                    ;fue cero, continuo
        btg     PORTE,0                     ;cambio el semiciclo positivo
        btg     PORTE,1                     ;cambio el semiciclo negativo
        movlw   0x24                        ;vuelvo a cargar el contador a
```

```
        movwf   CONTADOR1                       ;su valor original


        RETFIE      FAST
;*****************************************************************
;Lower priority interrupt service routine
;Change on PortB(Fault checking) interrupt & "ADConversion over"
;interrupt are checked
;*****************************************************************
ISR_LOW
        btfsc   INTCON,RBIF                     ;RB interrupt?
        bra     CHECK_FAULT                     ;Yes
        btfsc   PIR1,ADIF
        bra     AD_CONV_COMPLETE
        RETFIE      FAST
;*****************************************************************
CHECK_FAULT
        movf    PORTB,W                         ;Check for fault bit
        btfss   WREG,FAULT_BIT
        bra     THERE_IS_FAULT
        call    RUN_MOTOR_AGAIN                 ;Fault cleared?
        bcf     INTCON,RBIF                     ;Run motor again
        RETFIE      FAST
THERE_IS_FAULT                                  ;Yes,fault is there
        call    STOP_MOTOR                      ;Stop motor
        bcf     INTCON,RBIF
        RETFIE      FAST
;*****************************************************************
AD_CONV_COMPLETE                                ;ADC interrupt
        movff   ADRESH,FREQUENCY
        movlw   0x14                            ;Minimum Frequency set to 5Hz
                                                ;(scaling factor X4)
        cpfsgt  FREQUENCY
        movwf   FREQUENCY
        movlw   0xF0                            ;Limiting V/F to F= 60Hz
                                                ;(scaling factor X4)
        cpfslt  FREQUENCY
        movwf   FREQUENCY
        nop                                     ;Out frequency to PORTD
                                                ;for set value
        bcf     PIR1,ADIF                       ;ADIF flag is cleared for
                                                ;next interrupt
        RETFIE      FAST

;*****************************************************************
;This routine will update the PWM duty cycle on CCPx
```

```
;This routine scales the PWM value from the table based on the
;frequency to keep V/F constant.
;*****************************************************************************
UPDATE_PWM_DUTYCYCLES

        movf   TABLE_OFFSET2,W
        movf   PLUSW0,W
        bz     PWM2_IS_0
        mulwf  FREQUENCY
        movff  PRODH,CCPR2L_TEMP
        bra    SET_PWM12

PWM2_IS_0
        clrf   CCPR2L_TEMP                  ;Clear the PWM duty cycle register
        bcf    CCP2CON,4
        bcf    CCP2CON,5

SET_PWM12
        movff  CCPR2L_TEMP,CCPR2L
        bsf    PORT_LED1,LED1
        return




;*****************************************************************************
******
;This routine Updates the offset pointers to the table after every access
;*****************************************************************************
******
UPDATE_TABLE_OFFSET

        btfss  FLAGS,OFFSET2_FLAG           ;If set incr. on table
        bra    DECREMENT_OFFSET2
        movlw  (SINE_TABLE_ENTRIES-1)       ;Check for the last value on the table
        cpfslt TABLE_OFFSET2
        bra    CLEAR_OFFSET2_FLAG
        incf   TABLE_OFFSET2,F              ;Increment offset2

CLEAR_OFFSET2_FLAG
        bcf    FLAGS,OFFSET2_FLAG
DECREMENT_OFFSET2
        dcfsnz TABLE_OFFSET2,F              ;Decrement offset2
        bsf    FLAGS,OFFSET2_FLAG

        return
```

```
;*************************************************************************
******
;This routine calculates the Timer0 reload value based on ADC read value and the
;scaling factor calculated based on the main clock and number of Sine table entries.
;Timer0 value = FFFF - (FREQUENCY_SCALE/Frequency)       Frequ = (adc result)
;*************************************************************************
******
CALCULATE_FREQUENCY
        movff  FREQUENCY,TEMPORAL
        clrf   TEMP
        clrf   TEMP1
        movlw HIGH(FREQUENCY_SCALE)     ;FREQUENCY_SCALE/Frequency
        movwf TEMP_LOCATION             ;16 bit by 8 bit division
        movlw LOW(FREQUENCY_SCALE)
        movwf TEMP_LOCATION+1
continue_subtraction
        bsf    STATUS,C
        movf   FREQUENCY,W
        subwfbTEMP_LOCATION+1,F
        clrf   WREG
        subwfbTEMP_LOCATION,F
        btfss  STATUS,C
        goto   keep_result_in_rpm
        incf   TEMP,F
        btfsc  STATUS,C                 ;Result of the division is stored in
                                        ;TEMP&TEMP1
        incf   TEMP1,F
        goto   continue_subtraction
keep_result_in_rpm
;Timer0 value = FFFF-Timer0
        bsf    STATUS,C
        movlw 0xFF
        subfwbTEMP,F
        subfwbTEMP1,F                   ;The Timer0 reload value stored in
        movff  TEMP1,FREQ_REF_H         ;FREQ_REF_H & FREQ_REF_L
        movff  TEMP,FREQ_REF_L          ;These values will be loaded to
        return                          ;Timer0 in Timer0 overflow interrupt

;*************************************************************************
******
;This routine sets the ADC GO bit high after an aquisition time of 20uS approx.
;*************************************************************************
******
SET_ADC_GO
        call   CALCULATE_FREQUENCY
        btfss  ADCON0,GO
```

```
        bsf     ADCON0,GO               ;Set GO bit for ADC conversion start

        return

;*************************************************************************
******
;This routine initializes the parameters required for motor initialization.
;*************************************************************************
******

Init_Motor_Parameters

        clrf    CCPR2L                  ;Initialize all duty cycles to 0
        movlw   0x09
        movwf   TABLE_OFFSET2
        bcf     FLAGS,OFFSET2_FLAG
        movlw   0x30                    ;Initialize frequency to 12Hz
        movwf   FREQUENCY
        movlw   0xFD                    ;Timer0 Initialisation
        movwf   FREQ_REF_H
        movwf   TMR0H
        movlw   0x2C   ;
        movwf   TMR0L
        movwf   FREQ_REF_L
        bsf     FLAGS,TIMER0_OV_FLAG
        return

;*************************************************************************
******
;Upon initialization the Sine table contents are copied to the RAM from
;Program memory
;*************************************************************************
******
COPY_TABLE_TO_RAM
        movlw   UPPER sine_table        ;Initialize Table pointer to the first
        movwf   TBLPTRU                 ;location of the table
        movlw   HIGH sine_table
        movwf   TBLPTRH
        movlw   LOW sine_table
        movwf   TBLPTRL
        movlw   LOW(SINE_TABLE)
        movwf   FSR0L
        movlw   HIGH(SINE_TABLE)
        movwf   FSR0H
        movlw   0x14
        movwf   TEMP
COPY_AGAIN
```

```
        TBLRD*+
        movff  TABLAT,POSTINC0
        decfsz  TEMP,F
        bra     COPY_AGAIN

        movlw LOW(SINE_TABLE)                   ;FSR0 points to the starting
                                        ;of the table
        movwf FSR0L
        movlw HIGH(SINE_TABLE)
        movwf FSR0H
        return

;*************************************************************************
******
;This routine checks for the keys status. 2 keys are checked, Run/Stop and
;Forward(FWD)/Reverse(REV)
;*************************************************************************
******
KEY_CHECK
        btfsc   PORTA,RUN_STOP_KEY              ;Is key pressed "RUN/STOP"?
        bra     KEY_IS_RUN
        bcf     PORT_LED2,LED2
        call    STOP_MOTOR
        bsf     FLAGS,MOTOR_RUNNING
        return

KEY_IS_RUN
        bsf     PORT_LED2,LED2
        call    RUN_MOTOR_AGAIN
        bcf     FLAGS,MOTOR_RUNNING
        return


;*************************************************************************
******
;This routine stops the motor by driving the PWMs to 0% duty cycle.
;*************************************************************************
******
STOP_MOTOR

        clrf    CCPR2L
        clrf    TABLE_OFFSET2
        bcf     INTCON,TMR0IE
        bcf     PIE1,TMR2IE
        bcf     PIE1,ADIE
        bcf     PORTE,0
        bsf     PORTE,1
```

```
        return
;*****************************************************************************
******
;This routine starts motor from previous stop with motor parameters initialized
;*****************************************************************************
******
RUN_MOTOR_AGAIN
        call    Init_Motor_Parameters
        bsf     INTCON,TMR0IE
        bsf     PIE1,TMR2IE
        bsf     PIE1,ADIE
        return

;*****************************************************************************
******
;Rutina que despliega el valor de la frecuancia en codigo BCD
;*****************************************************************************
******
DESPLIEGA

        movff   TEMPORAL,FRECUENCIA     ;copio la parte alta de adresh
                                        ;a mi var frecuencia
        bcf     FRECUENCIA, 0           ;limpio los 2 primeros bits para
        bcf     FRECUENCIA, 1           ;después correr el byte 2 lugares
        rrncf   FRECUENCIA,1            ;a la derecha de manera que se divida
        rrncf   FRECUENCIA,1            ;entre 4 el byte, que es el factor
                                        ;que originalmente se uso
        movlw   0x09
        movwf   ACUMULADOR
        clrf    SUMADOR
BCD
        movf    ACUMULADOR,0            ;muevo acumulador al W
        cpfsgt  FRECUENCIA              ;comparo contra mi frecuencia
        goto    continua                ;es menor?, entonces continúa
        movlw   0x06                    ;debo sumar 6 al sumador
        addwf   SUMADOR,1               ;acumulo el valor al sumador
        movlw   0x0A                    ;muevo 10 para sumarlo al acumulador
        addwf   ACUMULADOR,1
        movlw   0x45                    ;llego al acumulador a 69?
        cpfseq  ACUMULADOR
        goto    BCD                     ;no? regreso a comparar



continua

        movf    SUMADOR,0
```

```
        addwf  FRECUENCIA,0
        movwf PORTD
        return

;*****************************************************************************
******
;Sine table for the inverter.
;*****************************************************************************
******
TABLE         code 0x0100
;below table is from 270 eg. to 90 deg @ 10 deg. resolution; for 20MHz,
;PR2 = F9, Timer2 1:1 prescale
sine_table db  0x0,0x03,0x08,0x11,0x1D,0x2D,0x3E,0x51,0x65,0x7B,0x90,
0xA5,0xB8,0xC9,0xD8,0xE5,0xEE,0xF3,0xF5
;*****************************************************************************
******



        END
```

ARCHIVO .INC

```
;User defined variables
;-------------------------------------------------------------------------------
;Oscillator frequency
#define OSCILLATOR        d'20000000'
;-------------------------------------------------------------------------------
;Timer0 prescaler
#define         TIMER0_PRESCALE         d'16'
;-------------------------------------------------------------------------------
;number of entries in the sine table, or the sampling frequency
#define         SINE_TABLE_ENTRIES      d'19'
;-------------------------------------------------------------------------------
SAMPLES_PER_CYCLE = (SINE_TABLE_ENTRIES-1)*d'2'
INSTRUCTION_CYCLE = (OSCILLATOR)/d'4'
FREQUENCY_SCALE =
(INSTRUCTION_CYCLE/SAMPLES_PER_CYCLE)/(TIMER0_PRESCALE/4)
;Timer prescale/4 is done to componsate ADC multiplication factor of 4 to the frequency)
;-------------------------------------------------------------------------------
;PWM frequency definition
#define TIMER2_PRESCALE         d'01'
#define         PWM_FREQUENCY           d'20000'
PR2_VALUE = (OSCILLATOR/(4*PWM_FREQUENCY*TIMER2_PRESCALE))-1
;-------------------------------------------------------------------------------
;ADC initialization
#define         Fosc_by_2       b'000'
#define         Fosc_by_8       b'001'
#define         Fosc_by_32      b'010'
#define         FRC             b'011'
#define         Fosc_by_4       b'100'
#define         Fosc_by_16      b'101'
#define         Fosc_by_64      b'110'

#define         ADC_CLOCK               Fosc_by_32
#define         ADC_CHANNEL             d'0'
#define         ADC_ON_BIT              b'1'
#define         LEFT_JUSTIFIED
#define         ADC_PORT_CONFIG         b'1110'         ;Refer the table in the manual
                                                        ;for selection

ADCON0_VALUE = ((ADC_CLOCK<<6)|(ADC_CHANNEL<<4)|(ADC_ON_BIT))

        if ((ADC_CLOCK==Fosc_by_2)||(ADC_CLOCK ==Fosc_by_8)||(ADC_CLOCK
==Fosc_by_32)||(ADC_CLOCK==FRC))
        ifndef  LEFT_JUSTIFIED
ADCON1_VALUE = ((1<<7) | (ADC_PORT_CONFIG))
        else
```

```
ADCON1_VALUE = ADC_PORT_CONFIG
        endif
        endif
        if
        ((ADC_CLOCK==Fosc_by_4)||(ADC_CLOCK==Fosc_by_16)||(ADC_CLOCK==
Fosc_by_64))
        ifndef  LEFT_JUSTIFIED
ADCON1_VALUE = ((1<<7) |(1<<6)| (ADC_PORT_CONFIG))
        else
ADCON1_VALUE = ( (1<<6)| (ADC_PORT_CONFIG))
        endif
        endif
;-----------------------------------------------------------------------------
;Port definitions
#define         DRIVER_ENABLE_PORT          PORTB
#define         DRIVER_ENABLE_BIT           3
#define         FAULT_BIT                   4
#define         FWD_REV_KEY                 4
#define         RUN_STOP_KEY                5
#define         PWM3_PORT                   PORTC
#define         PWM3_PORT_PIN               3
#define         PORT_LED1  PORTA
#define         LED1                        2
#define         PORT_LED2  PORTA
#define         LED2                        3
;-----------------------------------------------------------------------------
```