OVERVIEW

My idea for an electromechanical timer replacement is to use a PIC12C509 to count AC cycles to produce an accurate timer. The delay can be configured in software from 2 to 255 minutes by changing one variable. There is not user interface required. The device simply starts counting AC cycles on power up.

When the time has elapsed, the PIC12C509 triggers the output line which is connected to an optically isolated triac circuit. The PIC12C509 senses the start of the next cycle and powers the triac right on the zero crossing (this reduces wear and tear).

Ideally, this circuit was designed as a freezer protector. It counts 5 minutes from power up and then starts the freezer motor.

During short power failures, restarting the compressor in a freezer just after it has shut off increases wear and tear on the motor. The PIC12C509 prevents this by waiting 5 minutes, then restarting the freezer motor. This gives the refrigerant enough time to settle and therefore reduces load on the freezer motor. Waiting also prevents the high power drain caused by a large motor (the freezer) starting up from occurring at the same time as all the other circuits (if any) that are coming back on line. The reduces load on the power line on startup.

APPLICATION OPERATION

Hardware

The whole circuit is powered by the AC line. To obtain power for the PIC12C509 and related devices, a transformerless power supply is employed (see Microchip Technical Note TB008 (DS91008)).

The AC cycles are input right from the 117V AC line through a 4.7 Meg resistor which limits the current (see Microchip Application Note AN521 (DS00521)). The GPIO GP1 pin is configured and used as an input.

The circuit waits 5 minutes from power-up and then sets the GPIO GP0 line low. This fires the opto-triac (Motorola® MAC224A10 or any appropriate triac). The rest of the circuitry on the output provides the correct bias for the triac and opto-triac and includes a "snubbing" circuit. The snubbing circuit reduces inductive voltage transients such as those caused from a motor (such as a freezer motor) connected to this circuit. This improves the noise characteristics and the life of the triac. For more information on this circuit, please see Motorola Application Note AN780 (DS00780).

BLOCK DIAGRAM

```
AC In                       Current Limiting Resistor
                           5V Transformerless Supply
                           PIC12C509
                           Opto-Triac Circuit
                           Transient Snubber
                           Controlled AC Out
```

Author: Bret Walters
Inter.tec
Pickering State, Ontario
Canada
email: bretw@ibm.net
SOFTWARE

The software was designed for modularity. It was believed that the functions created were useful for other projects so it was decided that it would be best to break the functions into subroutines. It would be very easy to remove all of the subroutines and redo the code without them, but due to the nature of this design (speed not critical, 2 level hardware stack available), it is unnecessary.

The functions are discussed here:

• main - the main code
  This function controls the calling of all other functions and handles tasks like setting up the I/O lines and setting the output line low at the end of the program.

• Cycle Check
  This useful function checks the GPIO GP1 pin continuously. By watching for a high input and then a low input, the rising edge of the AC input is found. Plus, due to the speed of the PICmicro, this is the zero crossing as well. After this point is found, the function RETURNs control to main.

• Initialize Cycle Timer
  This function works with the function “Cycle Timer”. Calling it initializes all variables used in the “Cycle Timer” function to their original values. The values are set in the beginning of the code as a series of variables (using EQUates). The variables are cti1, cti2 and cti3. They are the 1/60 seconds, seconds and minutes initialization values respectively. “cti” stands for Count Initialize. The values initialize CTR1, CTR2 and CTR3 respectively, these are used for the function “Cycle Timer”.

• Cycle Timer
  This function is intended for use with the “Cycle Check” function but can be used anywhere. The function decrements 3 registers (CTR1, CTR2 and CTR3) in a count down fashion. It counts once updating the appropriate registers, then exits. In this submission the registers count 1/60s of a second, seconds and minutes.

  On exit, the function passes a value to the W register appropriate as to if the count is finished or not. The count is finished when the “minutes” counter reaches 0. If the count is not finished, a 1 is returned, signifying that the counter needs to continue. If the count is finished, a 0 is returned. The main code handles these conditions.

  The counting is accomplished by decrementing and looking for a 0 result. If a 0 is found, the 0 is reinitialized to the correct value and then the next variable is decremented. See the code and/or the Flow Chart for more details on this function.
FLOW CHART FOR AC TIMER SWITCH (FREEZER PROTECTION)

START

CLEAR GPI0

SET PINS AS 1 OR 0 IN GPI0

CALL INITIALIZE CYCLE TIMER

CALL CYCLE CHECK

CALL CYCLE TIMER CONTINUE?

N

CALL CYCLE CHECK

TURN ON OUTPUT GP0

END (LOOP CONTINUOUSLY)

Y

Y

Y

N

N

Y

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N

N
MICROCHIP HARDWARE DEVELOPMENT TOOLS USED

Initial code was tested on the PIC16C84 due to its convenient EEPROM configuration. Later versions were developed on the PIC12C509 - EPROM version. All parts were programmed using the PICSTART® Plus Development Programmer, part number: 10-00157 by Microchip Technology Inc. This was obtained through the complete PICSTART Plus Development System obtained from the 1997 Toronto conference.

Assembler/Compiler Version

MPLAB™ for Windows®/16 version 3.22.00 Microchip Technology Inc., running on Windows 95. Processor version 6.2, Disassembler version 2.0. Copyright Microchip Technology Inc. 1995. This was also obtained through the PICSTART Plus 1997 Toronto Conference.
APPENDIX A: SOURCE CODE

;----------------------INTRODUCTION-------------------------
;This code is by Bret Walters 1997. Version 0.9
;This is an unpublished work.
;
;This program is developed for the PIC12C509 design contest session
;2. It uses a 12C508/09 to protect an electrical device.
;The protection is that as an AC power failure occurs, all
;devices attempt to turn on at once, causing havoc with the AC
;line.
;
;Devices like freezers and the like could be turned on
;significantly after the restoring of power. This example
;uses a 5 minute delay.
;
;The following code was designed for the 12C509 (the part I
;had on hand) and has been successfully simulated using
;MPLAB 3.22.00

;----------------------FUNCTION-----------------------------
;This code counts the number of AC cycles which are 1/60 s
;apart (or 1/50 s in Europe - a simple code change), via
;GP1. Pulses enter via the AC line through a 4.7 Meg resistor
;to the PIC12C509 (at GP2). Output from the PIC12C509 at GP0 drives an
;optocoupler which drives a triac that turns on the electrical
;device. A solid state relay could also replace the
;triac/optocoupler combination.
;
;Zero crossings (in the rising edge) are counted providing an
;accurate timing source. This is done by looking for a low
;then looking for a high input. One of the best features of
;this design is that the device is only turned on at the zero
;crossing, reducing noise and eliminating the need for a zero
;crossing circuit

;---------------------ACKNOWLEDGMENTS-----------------------
;The power supply used is featured
;in the Microchip Application Notes. The Triac/Optocoupler
;configuration is a design of Motorola Inc.

;---------------------FUTURE ENHANCEMENTS-------------------
;Using careful timing would allow the PIC12C509 to turn on the
;triac at different points on the AC cycle
;(since the zero crossing point is known) to provide a
;dimmer for other circuits.

;----------------------PART--------------------------------
;list p=12c509
;include "p12c509.inc"
;include "p12c509.inc"

ScratchPadRam   equ   0x07
CTR1           equ   ScratchPadRam+0
CTR2           equ   ScratchPadRam+1
CTR3           equ   ScratchPadRam+2
CHECK1         equ   ScratchPadRam+3
w              equ   .0
f              equ   .1
z              equ   .2
PORTA          equ   0x06
STATUS         equ   0x03
cti1           equ   .60
cti2           equ   .60
cti3           equ   .5
FALSE          equ   .0
TRUE           equ   .1
;-----------------------------------CODE-----------------------------------
    org 0 ;start address 0
    clr PORTA ;clear output to prevent error and noise on powerup
    bsf PORTA,0 ;set appliance to off state
    movlw 0x1E ;set all ports
    tris PORTA ; as inputs except GP0
    call icytmr ;call initialize cycle timer for cyctmr routine
    call cycchk ;call cycle check, look for 1 AC cycle and return
    call cyctmr ;call cycle timer, count to 5 minutes and exit
    movf CHECK1,F ;set flags appropriate for CHECK1
    btfss STATUS,z ;a 0 returned from cyctmr will end the counting
    goto cycchk ;call cycle check, after 1 AC cycle,
    bcf PORTA,0 ;turn on appliance
    stphre goto stphre ;stop here

;***CYCLE CHECK - Get rising edge of AC cycle on GP1***
cycchk bitsc PORTA,1 ;check AC input at GP1 for 0
    goto cycchk ;check AC input at GP1 for 1
    low_ok btfss PORTA,1
    goto low_ok ;check AC input at GP1 for 1
    return ;found zero crossing - rising edge

;***INITIALIZE CYCLE TIMER - Initializes cyctmr***
icytmr movlw cti3 ;initialize
    movwf CTR3 ; counter 3 (counts minutes)
    movlw cti2 ;initialize
    movwf CTR2 ; counter 2 (counts seconds)
    movlw cti1 ;initialize
    movwf CTR1 ; counter 1 (counts 1/60ths of a second)
    return

;***CYCLE TIMER - counts calls since initialization to create a specific delay time***
cyctmr decfsz CTR1,F ;nest 1:Decrements preloaded value
    retlw TRUE ; to 0 and then,
    movlw cti1 ;initializes
    movwf CTR1 ; counter 1 (counts 1/60ths of a second)
decfsz CTR2,F ;nest 2: Decrements preloaded value
    retlw TRUE ; to 0 and then,
    movlw cti2 ;initializes
    movwf CTR2 ; counter 2 (seconds)
decfsz CTR3,F ;nest 3: Decrements preloaded value
    retlw FALSE ; exits (0)
end