**A Simple Programmable Timer with Time Correction Circuit**

**OVERVIEW**

The timer discussed in this application note may be used to operate different appliances using AC or DC voltage (battery powered equipment). It is adjustable from 1 second to 99 hours, 59 minutes, 59 seconds and has a time correction circuit to adjust the accuracy of the internal 4 MHz RC oscillator. It is based on the PIC12C508 microcontroller and requires few external components.

The system block diagram is shown in Figure 1. The timer has 3 buttons, an LCD panel, and a power switch. The power switch, in the position shown in Figure 1, will activate the timer, and in the other position the load will be connected directly to the line out, thus switching off the opto-transformer to prevent accumulator overcharge. When switched off, the timer is still operational and may be set. To reduce power consumption, the timer goes to sleep (LCD panel appears blank) in 15 seconds after the last button depression, while in edit mode.

The first keyboard button is “Change Digit.” This button is used to position the cursor to the desired digit. The cursor moves clockwise from the lower seconds digit to the higher hours digit. Separators are skipped. The second button is “Increase Value.” With each depression of this button the digit value is increased. After 9 presses the value rolls over to 0. The third button is “Start/Stop.” This button is used to edit data and to activate the timer. When run in edit mode, with the earlier specified buttons, data may be entered. Once the Start/Stop button is pressed, the timer is activated. It stops if the time expires or the Start/Stop button is pressed again (in both cases this will affect the load). In any case, if the timer is stopped, then the edit mode becomes active and the remaining data may be edited.

This appliance may be used as a simple timer and if used with a normally closed contact opto-relay, it may turn on any load when the time expires.

Use an 8-pin socket for the PIC12C508 JW device because it needs to be programmed before putting it in the circuit. The board has three buttons (S1, S2, S3), capacitor (C2), resistor (R1), and an LED (instead of an opto-relay).

The opto-relay may be replaced with an opto-triac (AC operation only). Using a triac or an ordinary mechanical relay with an opto-relay or an opto-triac will power-up the output (DS00559). The opto-relay may be replaced with an ordinary mechanical relay, but most of them need a higher voltage and a transistor amplifier to activate their coil.
HARDWARE

The schematic diagram is shown in Figure 2.

FIGURE 2: SCHEMATIC DIAGRAM

![Diagram of the circuit](image)

**Note 1:** GND symbol is circuit common; it is not tied to AC ground.

2: In case of DC input voltage, C4 should be replaced with a resistor to limit current flow through the LEDs (R3 may be removed).

3: If the input voltage is 36V or less, the opto-transformer may be removed.

4: DISPLAY means AY0438 IC with LCD panel added.

5: A FUSE is required in-Line In-circuit.

The Timer Parts List

**Capacitors:**

C1, C3 = 47 μF electrolytic
C2 = 0.1 μF ceramic
C4 = 1 μF (depends on the current flow limits through LEDs)

**Diodes:**

D1 – D6 = Any type IR photo diodes
D7 – D12 = Any type IR light emitting diodes
D13 = 1N4001 general purpose 1A rectifier

**Resistors:**

R1 = Depends on the type of opto-relay:
\[ V_O = \text{PICmicro™ output low voltage (0.6V max)} \]
\[ V_{LED} = \text{Input opto-relay IR LED voltage (0.8V typical)} \]
\[ I_{LED} = \text{LED current (10 mA typical)} \]
\[ R1 = \frac{3 - V_{LED} - V_O}{I_{LED}} = 160\Omega \]

R2 = 100Ω (depends on the current flow limits through transformer LEDs and output voltage of photo diodes battery to provide minimal charge current for the battery)

R3 = 1 MΩ, used to discharge capacitor C4

**Miscellaneous:**

Battery = 3V Accumulator
Opto-relay = Any opto-relay or opto-triac matching above specified requirements
S1 – S3 = Normally open push button switches
S4 = Two position switch
U1 = PIC12C508 programmed with MyTimer code

MICROCHIP TOOLS USED

Assembler/Compiler Version:
MPASM version 1.40
APPENDIX A: SOURCE CODE

;A Simple Timer with Time Correction Circuit
;Author: Kirill Yelizarov

LIST P= PIC12C508, R= DEC
INCLUDE <p12c508.inc>

__CONFIG _HS_OSC & _WDT_OFF & _CP_OFF & _MCLRE_OFF

; ------------ DATA ------------
TimerStatus equ 0x07 ;Timer status flags
SecondsLow equ 0x08 ;Seconds low digit
SecondsHigh equ 0x09 ;Seconds high digit
MS_Separator equ 0x0a ;Minutes/Seconds separator
MinuteLow equ 0x0b ;Minutes low digit
MinuteHigh equ 0x0c ;Minutes high digit
HM_Separator equ 0x0d ;Hours/Minutes separator
HourLow equ 0x0e ;Hours low digit
HourHigh equ 0x0f ;Hours high digit
TimerCount equ 0x10 ;Timer count
Digit equ 0x11 ;Used in ShowDigit to send data using RRF command
Count equ 0x12 ;Multipurpose count
TimePatch equ 0x13
TimeCorrection equ 0x14

; ------------ Timer Status Flags ------------
SleepFlag equ 7 ;If set then the sleep command is activated
SetTimeFlag equ 6 ;Flag is raised when it's time to update time
StartStopFlag equ 5 ;If raised the timer runs else it may be edited
TimerFlag equ 4 ;this flag is raised when TMR0 seventh bit is set
; equ 3
; equ 2
; equ 1
; equ 0

; ------------ Keyboard & LCD hardware bits ------------
LCD_Clock equ 5 ;LCD clock
LCD_Data equ 4 ;LCD data
StartStop equ 3 ;Start/Stop button
Relay equ 2 ;Relay
IncValue equ 1 ;Increase digit value button
NextDigit equ 0 ;Set next digit button

; ------------ CODE ------------
org 0
clrf TimerStatus ;Reset all flags
goto ResetTimer ;Reset timer and read time correction value

; Decode data to 8 segment LCD
DecodeValue addwf PCL,F
retlw b'00011111' ;0
retlw b'00000110' ;1
retlw b'01011011' ;2
retlw b'01001111' ;3
retlw b'01100110' ;4
retlw b'01101101' ;5
retlw b'01111101' ;6
retlw b'00000111' ;7
Electromechanical Timer Replacement

; Time correction table
; In correction mode the timer outputs 250 kHz signal on pin LCD_clock
; This may be tested and a new correction value be programmed
; If bit <6> is zeroed then the next value is fetched (because 1’s may be programmed many times)
; TimeCorrectionTable

    ;bit <7> 1 = Correction mode                       ; 0 = Normal timer operation
    ;bit <6> 1 = Read this value                       ; 0 = Skip this value
    ;bits <5,0> Time correction value

    retlw b'11000111' ; Set time correction value to 7
    retlw b'11111111'
    retlw b'11111111'
    retlw b'11111111'
    retlw b'11111111'
    retlw b'11111111'
    retlw b'11111111'
    retlw b'11111111'

; ------------------- SUBROUTINES -------------------
; Send timer data from lowest digit to the highest including separators to AY0438
ShowTime

    bcf TimerStatus, SetTimeFlag ; reset SetTimeFlag bit
    movlw 0x0a
    movwf FSR ;get first digit (seconds low)
    ShowNextDigit:

    movf INDF, W
    call DecodeValue
    btfsc TimerStatus, SleepFlag ;If Sleep mode activated
    movlw 0x10
    ; then set display blank
    call ShowDigit
    incf FSR, F ; get next digit
    btfss FSR, 4 ; terminate if the 4th bit is set
    goto ShowNextDigit ;(You should not move timer data in RAM
    return ; for this function to work properly)

; Transmit one digit data to AY0438 (see PIC16/17 Microcontroller Data Book 95/96 AY0438 page 4-6)
; Load input should be tied high
ShowDigit

    movwf Digit ; Save current digit value
    movlw 0x08
    movwf Count ; Set Count to 8 (8 segment LCD)

NextBit:

    bcf GPIO, LCD_Data ; clear Data bit
    rrf Digit, F
    btfsc STATUS, C ; If bit is clear then skip
    bsf GPIO, LCD_Data ; Else set Data bit
    bsf GPIO, LCD_Clock ; Toggle Clock, the data to be read by AY0438
    bcf GPIO, LCD_Clock
    decfsz Count, F
    goto NextBit
    return

; Get current time from TMR0
GetTime
Electromechanical Timer Replacement

btfss    TMR0,7            ;atch for the TMR0 seventh bit getting set
return
btfsc    TimerStatus,TimerFlag     ;and TimerFlag is not set
return
bsf      TimerStatus,TimerFlag     ;then set this flag, this will prevent from coming
        ;here again till the seventh bit is reset and set again
decfsz   TimerCount,F
return
bsf      TimerStatus,SetTimeFlag  ;Set the flag to update time value
movlw    30
btfsc    SecondsLow,0            ;add to odd values extra count to increase accuracy
movlw    31
movwf    TimePatch
decfsz   TimePatch,F             ;This patch will increase accuracy
        ;which is quite suitable for a timer
movlw    59                      ;After every 59th second an extra count
movwf    TimePatch               ;will be assigned to timer count
incf     TimerCount,F
return

;                 ------------- Reset Timer and Main Loop -------------
ResetTimer:
movlw    0x0a
movwf    HM_Separator            ;set Hours/Minutes separator to 11
movlw    0x0b
movwf    MS_Separator            ;set Minutes/Seconds separator to 12
bcf      TimerStatus,SetTimeFlag ;reset SetTimeFlag bit to avoid mistakes
bsf      TimerStatus,TimerFlag   ;to run a complete first second
movlw    b'10000000'             ;timer will start from the value of 128
movwf    TMR0         ;and the time will be updated every time the seventh bit is set
movlw    b'00000110'             ;set prescaler to 1:128
option
movlw    0xff                  ;set Count to 255
movwf    Count
NewValue:
incf     Count,F                ;this will make Count = 0 for the first fetch
call     TimeCorrectionTable
movwf    Digit                  ;save time correction value
btcss    Digit,6                ;if the value is valid (bit <6>)
goto     NewValue               ;if not get the next one
andlw    b'00001111'
movwf    TimeCorrection
swapf    TimeCorrection,F
btfsc    Digit,7                ;analyze mode
        ;skip for the normal operation
Main:
btfsc    TimerStatus,StartStopFlag
call     GetTime                ;Change time if StartStopFlag is set
btfsf    TimerStatus,SetTimeFlag
goto     Main
decf     SecondsLow,F
call     ShowTime               ;Update time if SetTimeFlag is set
goto     Main
CorrectionSignal:                ;250 kHz correction signal
bsf      GPIO,LCD_Clock
nop
nop
bsf      GPIO,LCD_Clock
nop
goto     CorrectionSignal
org      0x1ff
movlw    b'01110000'
end