



Electromechanical Timer Replacement

99 Minute Timer

*Author: Milind Hareshwar Patil
Tata Institute Of Fundamental Research
Mumbai, Maharashtra
India
email: milindh@tifrvax.tifr.res.in*

OVERVIEW

This 99 minute timer replaces the electro-mechanical timer. The hardware is made simple and economical. The timer is very simple to operate. Two 7-segment displays are provided to display minutes from 0 to 99. To simplify operation, only two push-on switches are provided to set minutes. Each switch is dedicated to each display.

“TENS OF MIN SW” increments tens of minutes display from 0 through 9 at each push and “UNITS OF MIN SW” increments units of minutes displays from 0 through 9 at each push. Debouncing of switches is taken care by software.

APPLICATION OPERATION

The TMR0 prescaler is set to x16. Software checks TMR0 for 250 rollovers, which happens every 4 msec. 4 msec is counted 25 times to generate 100 msec, from which seconds and minutes are generated.

Every 100 msec the keyboard is read and the required minutes display (TENS or UNITS) is incremented at the rate of 2 Hz.

If the displayed minutes is equal to 0 then relay is put off and UNITS DISPLAY's decimal point is not flashing to indicate the timer is off.

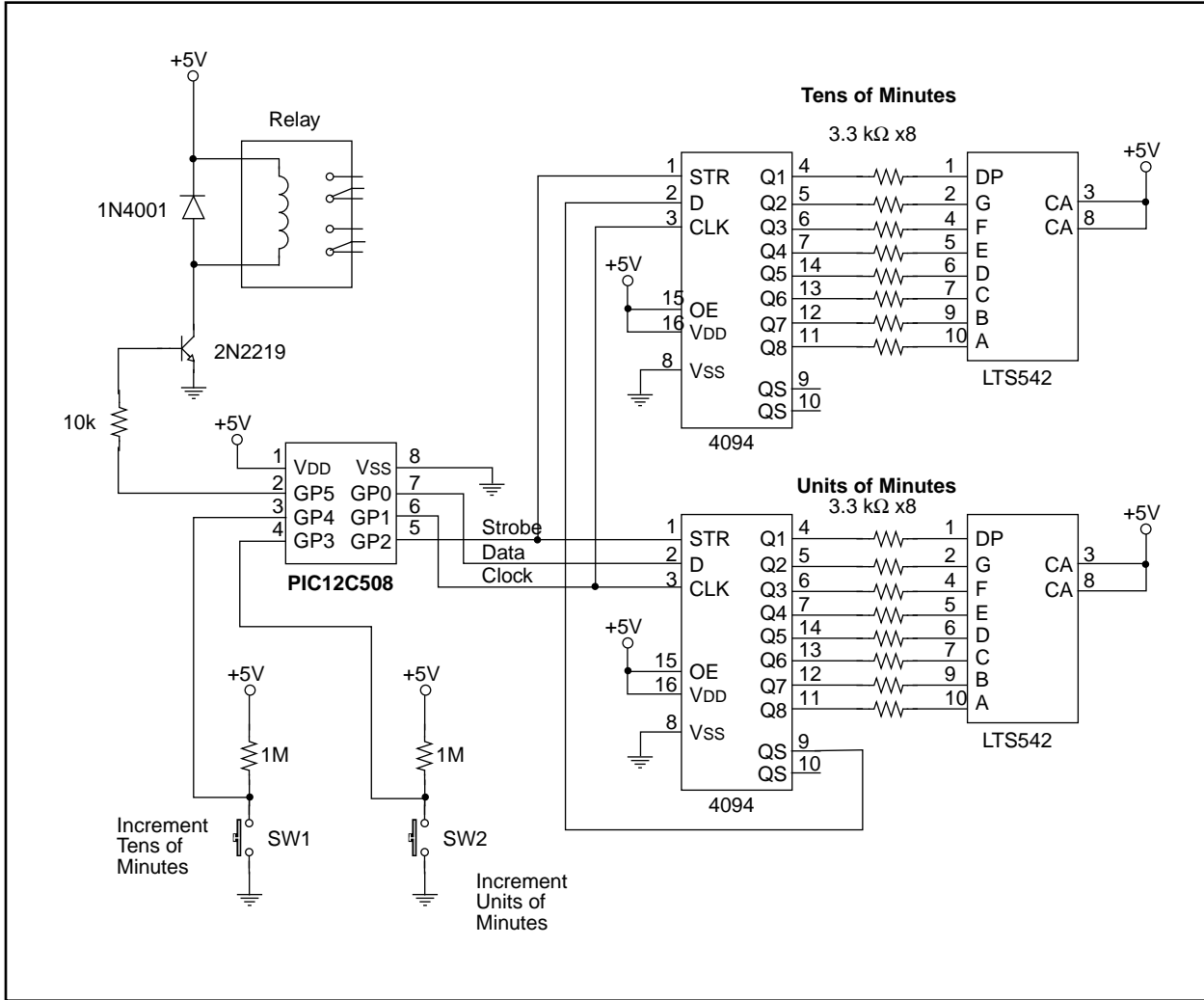
If displayed minutes is not equal to 0, then a relay is turned on and UNITS DISPLAY's decimal point is flashed to indicate the timer is active. During this mode the display decrements every minute till it reaches 0, where the relay is put off.

- Note 1:** Total instruction time [one loop]:
Approximately 375 μ sec, maximum.
- 2:** Set Processor configuration word as
0000 0000 1010b.
- $\overline{\text{MCLR}}$ tied to VDD (internally).
 - Code protection off.
 - WDT disabled.
 - Internal RC oscillator.

Microchip Technology Incorporated, has been granted a non-exclusive, worldwide license to reproduce, publish and distribute all submitted materials, in either original or edited form. The author has affirmed that this work is an original, unpublished work and that he/she owns all rights to such work. All property rights, such as patents, copyrights and trademarks remain with author.

Electromechanical Timer Replacement

FIGURE 1: SCHEMATIC



MICROCHIP TOOLS USED

Assembler/Compiler Version

MPASM 1.50 released.

Electromechanical Timer Replacement

APPENDIX A: SOURCE CODE

```
;          define ram ports and data ;          ***** ;
DEFINE PORT
;          *****
;DEFINE PORTS
;*****
gp0      equ      0
gp1      equ      1
gp2      equ      2
gp3      equ      3
gp4      equ      4
gp5      equ      5
;DISPLAY PORT
;*****
disp_data equ      gp0
disp_clk  equ      gp1
disp_strobe equ    gp2
;KEYBOARD PORT
;*****
;units_key equ      gp3
;tens_key  equ      gp4
;RELAY PORT
;*****
relay    equ      gp5
;*****
;DEFINE RAM
units    equ      08h          ;minutes unit display
tens     equ      units+1      ;minutes tens display
counter_4ms equ    tens+1      ;incremented every 4 msec
counter_100ms equ  counter_4ms+1 ;incremented every 100 msec
seconds  equ      counter_100ms+1
oldkey   equ      seconds+1    ;for key debouncing
newkey   equ      oldkey+1     ;----- ,, -----
key      equ      newkey+1     ;key pressed data
digit_inc equ    key+1         ;inc units display if bo = 1
                                ;inc tens display if b1 = 1
tmr_comp equ    digit_inc+1    ;tmr0 comparator register
scrtch0  equ      tmr_comp+1
scrtch1  equ      scrtch0+1
scrtch2  equ      scrtch1+1
;*****
;DEFINE FLAGS AND BITS
b0      equ      0
b1      equ      1
b2      equ      2
b3      equ      3
b4      equ      4
b5      equ      5
b6      equ      6
b7      equ      7
;key
;***
;          equ      b0
;          equ      b1
;          equ      b2
units_key equ    b3
tens_key  equ    b4
;          equ      b5
;          equ      b6
;          equ      b7
;digit_inc
;*****
units_inc equ    b0
tens_inc  equ    b1
;          equ      b2
```

Electromechanical Timer Replacement

```
;          equ      b3
;          equ      b4
;          equ      b5
;          equ      b6
;          equ      b7
;*****
#define data_hi      bsf      GPIO, disp_data
#define data_lo      bcf      GPIO, disp_data
#define clk_hi       bsf      GPIO, disp_clk
#define clk_lo       bcf      GPIO, disp_clk
#define strobe_hi    bsf      GPIO, disp_strobe
#define strobe_lo    bcf      GPIO, disp_strobe
#define relay_on     bsf      GPIO, relay
#define relay_off    bcf      GPIO, relay
;***** EOF DEF_RAM.ASM *****
;          SUBS.ASM
;          *****
;read key port in key
read_keys          movf      newkey, w          ;debouncing taken care by s/w
                  movwf     oldkey
                  movf      GPIO, w           ;read newkeys
                  andlw     00011000b        ;gp3, gp4 are key ports
                  movwf     newkey
                  comf      newkey, f        ;complement since active low
                  andwf     oldkey, w        ;key=oldkey AND compl(newkey)
                  movwf     key
;indicate to display routine units or tens key pressed
                  btfsc    key, units_key
                  bsf      digit_inc, units_inc
                  btfsc    key, tens_key
                  bsf      digit_inc, tens_inc
                  return
;*****
init_ports         movlw     0                ;all lo
                  movwf     GPIO
                  movlw     00011000b        ;port g0-g2,g5 o/p & gp3,gp4 i/p
                  tris      GPIO
                  clrf      TMR0            ;clr tmr0 & prescaler
                  movlw     11000011b        ;tmr0 enable with 1:16 prescaler
                  option
                  return
;***** EOF SUBS.ASM *****
;          TABLES.ASM
;          *****
;7 segments decoded data. lo is segment on & hi is segment off
get_seg           addwf     PCL, f
;
;          ABCDEFGP          ;P is decimal point
retlw  00000011b    ;0
retlw  10011111b    ;1
retlw  00100101b    ;2
retlw  00001101b    ;3
retlw  10011001b    ;4
retlw  01001001b    ;5
retlw  01000001b    ;6
retlw  00011111b    ;7
retlw  00000001b    ;8
retlw  00011001b    ;9
dec_pt           equ      1111110b          ;decimal point bit
;***** EOF TABLES.ASM *****
```

Electromechanical Timer Replacement

```
;          TIMER.ASM
;          *****
; [ milindh@tifrvax.tifr.res.in ]
;Set Processor configuration word as = 0000 0000 1010 b.
; a) -MCLR tied to VDD (internally).
; b) Code protection off.
; c) WDT disabled.
; d) Internal RC oscillator [4 MHZ].
list      p=12c508, r=dec
          include "d:\pic\mpasm\p12c508.inc"

#define ram
          include "def_ram.asm"
;processor start
          org      0
          goto     start
;define legends and table
          include  "tables.asm"
;subroutines
          include  "subs.asm"
;initialize and start
start
          clrf     key
          clrf     oldkey
          clrf     newkey
          clrf     digit_inc
          clrf     units
          clrf     tens
          clrf     counter_4ms
          clrf     counter_100ms
          clrf     seconds
          movlw    250           ;4000us tmr0 (1:16 prescaler * 250 )
          movwf    tmr_comp
          call     init_ports    ;init ports & timer
main
          movf     tmr_comp, w   ;is tmr0 = tmr_comp (4 msec over?)
          xorwf    TMR0, w
          btfss   STATUS, Z      ;skip if = 250
          goto     main
          movlw    250
          addwf    tmr_comp, f   ;update 4 msec compare register
          incf     counter_4ms, f
          movlw    25
          xorwf    counter_4ms, w ;is 4ms * 25 = 100ms over ?
          btfss   STATUS, Z      ;skip if = 100ms
          goto     main
          clrf     counter_4ms
;***** 100 MSEC OVER *****
          incf     counter_100ms, f
          movlw    10
          xorwf    counter_100ms, w ;is 100ms * 10 = 1sec over ?
          btfss   STATUS, Z      ;skip if 1 sec over
          goto     main21        ;jmp if 1 sec not over
          clrf     counter_100ms ;1 sec over
          incf     seconds, f
          movlw    60
          xorwf    seconds, w    ;is 1 minute over ?
          btfss   STATUS, Z      ;skip if 1 min over
          goto     main21
;***** 1 MINUTE OVER *****
          clrf     seconds
;process relay and display
;if tens & units both = 0 then rly off and out
;else relay on & decrement display & out
          movf     tens, w
          iorwf    units, w
          btfss   STATUS, Z
          goto     main1
          relay_off
```

Electromechanical Timer Replacement

```
        goto    main21
main1    relay_on
        movf    units, f
        btfss   STATUS, Z
        goto    main2
        movlw   9
        movwf   units
        decf    tens, f
        goto    main21
main2    decf    units, f
;***** EXECUTED EVERY 100 MSEC *****
main21   call    read_keys
;** DISPLAY ROUTINE TO BE EXECUTED EVERY 500 MSEC **
        ;process display every 500 msec
        ;if unit key pressed increment unit display
        ;if tens key pressed increment tens display
        ;flash unit display's dec. pt. every second if units & tens
        ;not equal to zero i.e. indicate timer is active
        ;execute display routine in counter_100ms = 0 or 5
        movf    counter_100ms, w
        btfsc   STATUS, Z
        goto    main3
        xorlw   5
        btfss   STATUS, Z
        goto    main
        ;if units_key pressed then inc units display between 0 thr. 9
        ;if tens_key pressed then inc tens display between 0 thr. 9
main3    btfss   digit_inc, units_inc
        goto    main4
        bcf     digit_inc, units_inc
        ;inc units display. if units = 10 then units = 0
        incf    units, f
        movlw   10
        xorwf   units, w
        btfsc   STATUS, Z          ;skip if not equal
        clrf    units
main4    btfss   digit_inc, tens_inc
        goto    main5
        bcf     digit_inc, tens_inc
        ;inc tens display. if tens = 10 then tens = 0
        incf    tens, f
        movlw   10
        xorwf   tens, w
        btfsc   STATUS, Z          ;skip if not equal
        clrf    tens
        ;convert decimal in units & tens to decoded segment data
        ;for led display in scrтч1 & 2 for transmission
main5    movf    tens, w
        call    get_seg
        movwf   scrтч1
        movf    units, w
        call    get_seg
        movwf   scrтч2
        ;flash (toggle) decimal point every 500ms if timer active
        ;i.e. if timer not equal to zero i.e. relay on, then flash
        ;decimal point else, do not to indicate relay off
        movf    tens, w
        iorwf   units, w
        btfsc   STATUS, Z          ;skip to flash dp
        goto    main6             ;jmp if 0 to no flash
        movf    counter_100ms, f
        btfss   STATUS, Z          ;toggle-on dec pt if counter_100ms = 0
        goto    main6             ;else jmp out to toggle-off
        ;flash dp of units display
        movlw   dec_pt
        andwf   scrтч2, f
```

Electromechanical Timer Replacement

```
        ;transmit data from scrтч1 thr scrтч1 to display circuit (msb first).
main6   movlw  16                ;no of bit to tx
        movwf  scrтч0
main61  rlf    scrтч2, f         ;check msb
        rlf    scrтч1, f
        btfsc  STATUS, C       ;data hi if cy=1 else lo
        goto  main62
        data_lo
        goto  main63
main62  data_hi
main63  nop                    ;delay
        nop
        clk_hi                 ;toggle clk to push data
        nop
        nop
        clk_lo
        nop
        nop
        decfsz scrтч0, f        ;next bit
        goto  main61
        strobe_hi              ;strobe data nop
        nop
        strobe_lo
        data_lo                ;leave data lo
        goto  main
        end
; ***** EOF MAIN.ASM *****
```



Electromechanical Timer Replacement

Darkness Controller for Poultry

Author: Jerome Knapp
 Searle Research
 Skokie, IL
 USA
 email: jknapp@skcla.monsanto.com

OVERVIEW

Laying hens are very sensitive to the number of hours of light that they receive each day. The light has a direct affect on their hormonal systems and hence egg production. Ideal lighting conditions would never allow them to experience a decrease in daily light. This, in effect, means that their nights should be consistently as short as the shortest night of the year.

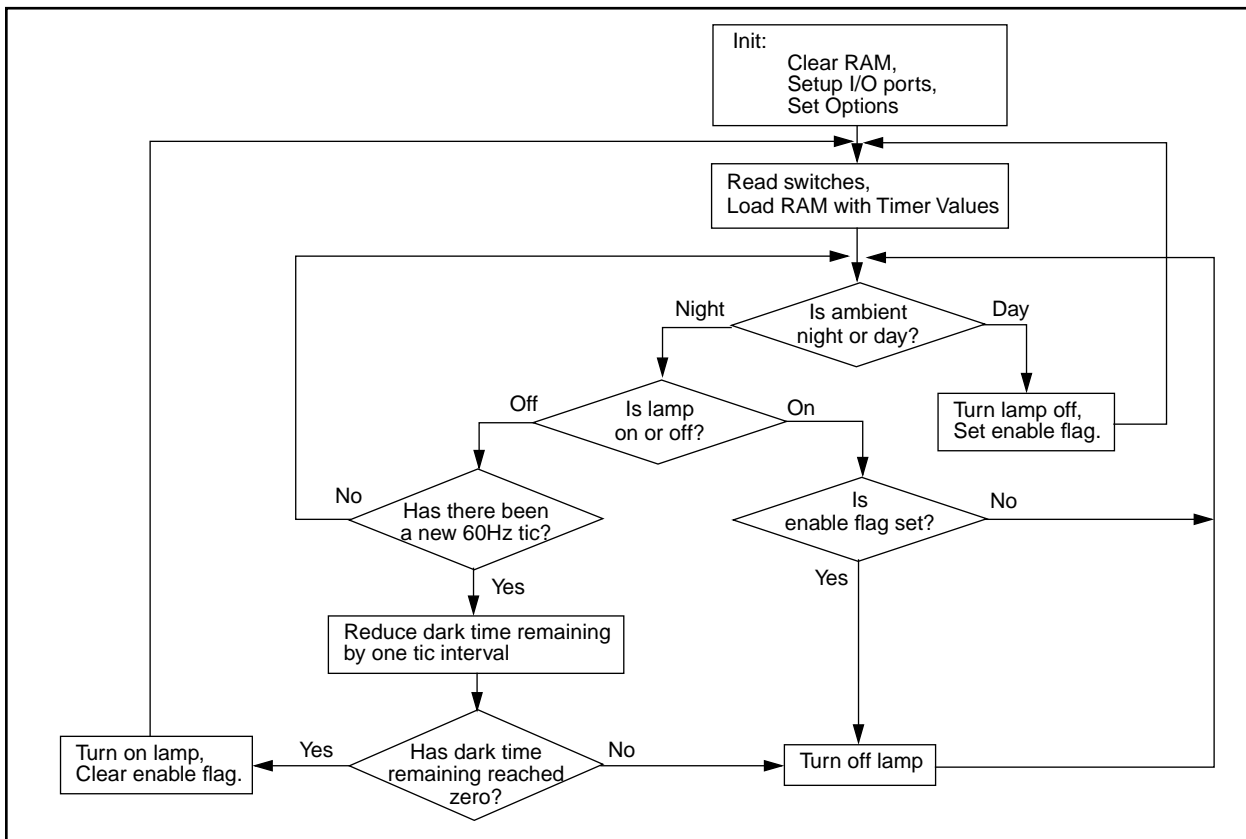
Most poultrymen use a simple synchronous motor-driven timer to put the lights on before dawn. This requires frequent adjustment if the hours of darkness

are to remain constant, since both the time of sunset and sunrise are constantly changing. An alternative is to leave the lights on all the time -- a waste of energy.

The subject of this application note is a "Darkness Controller" which senses the onset of twilight, delays for an amount of time, corresponding to the local shortest night, then switches on the lights in the poultryhouse. At dawn, the increase in natural light is sensed and the lights are turned off.

Line frequency is used as a time-base, since the controller may be subject to temperature extremes in an outdoor environment which may cause the internal oscillator to drift. A CdS photo-resistive cell in a voltage divider network is used to sense changes in the ambient light levels. (Must be shielded from the light being controlled). The light being controlled is switched by a Triac. Darkness duration is selected by dip switches. Four intervals are available, 7, 7.5, 8.0, and 8.5 hours.

FIGURE 1: FLOW DIAGRAM



Electromechanical Timer Replacement

Two switches allow selection of four darkness durations:

7.0 hours=1,512,000sixty hertz tics (171240 hex)

7.5 hours=1,620,000sixty hertz tics (18b820 hex)

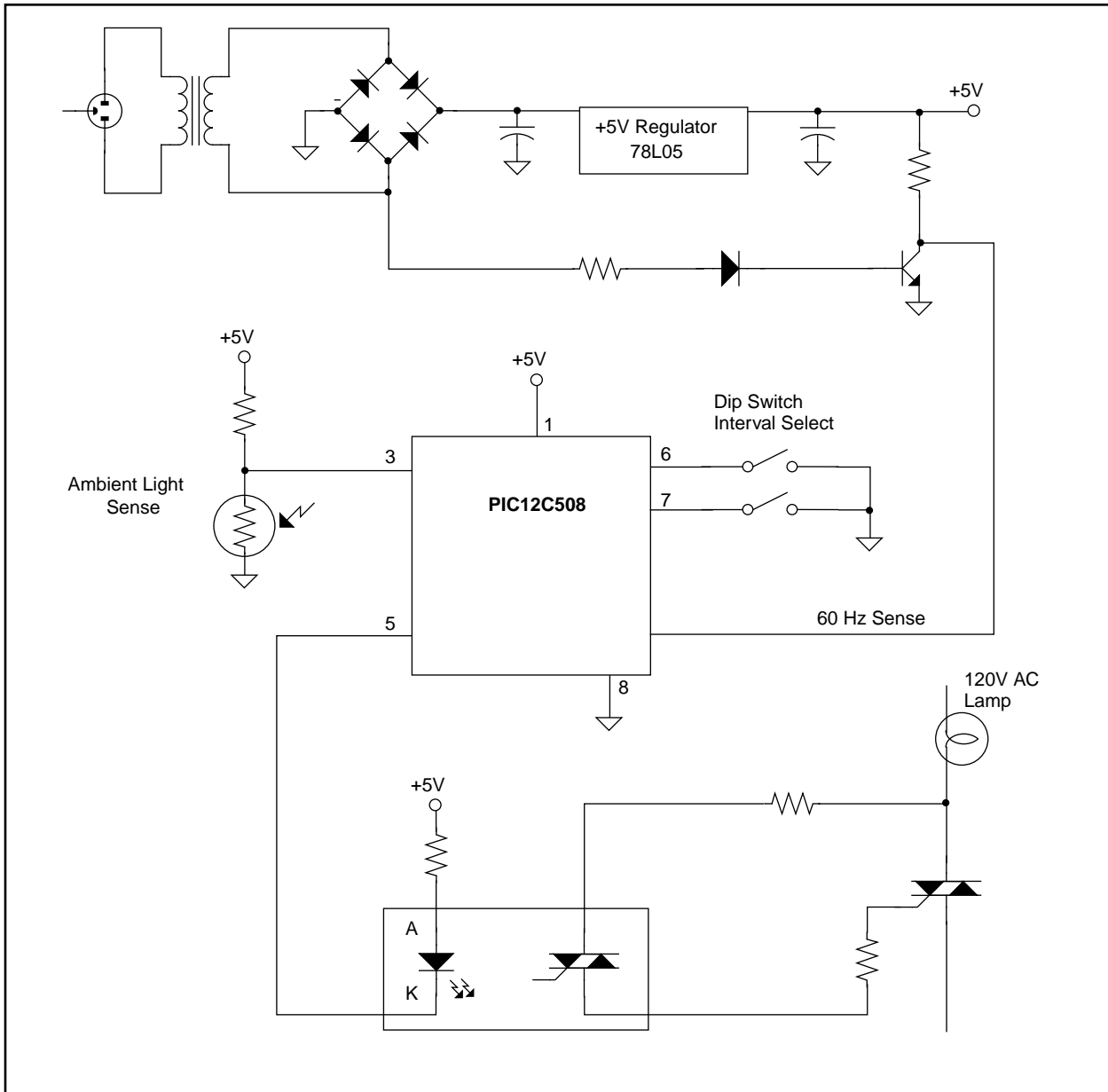
8.0 hours=1,728,000sixty hertz tics (1A5E00 hex)

8.5 hours=1,836,000sixty hertz tics (1C03E0 hex)

These hex values are loaded into 3 bytes of RAM and are decremented at each 60 Hz tic interval.

The enable flag prevents the lamp from being turned off. When it is still dark, nighttime after the required interval of darkness has elapsed.

FIGURE 2: SCHEMATIC DIAGRAM POULTRY DARKNESS CONTROLLER





MICROCHIP

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office

Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602-786-7200 Fax: 602-786-7277
Technical Support: 602 786-7627
Web: <http://www.microchip.com>

Atlanta

Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston

Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508-480-9990 Fax: 508-480-8575

Chicago

Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas

Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 972-991-7177 Fax: 972-991-8588

Dayton

Microchip Technology Inc.
Two Prestige Place, Suite 150
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Los Angeles

Microchip Technology Inc.
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 714-263-1888 Fax: 714-263-1338

New York

Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516-273-5305 Fax: 516-273-5335

San Jose

Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto

Microchip Technology Inc.
5925 Airport Road, Suite 200
Mississauga, Ontario L4V 1W1, Canada
Tel: 905-405-6279 Fax: 905-405-6253

ASIA/PACIFIC

Hong Kong

Microchip Asia Pacific
RM 3801B, Tower Two
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2-401-1200 Fax: 852-2-401-3431

India

Microchip Technology Inc.
India Liaison Office
No. 6, Legacy, Convent Road
Bangalore 560 025, India
Tel: 91-80-229-4036 Fax: 91-80-559-9840

Korea

Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Shanghai

Microchip Technology
RM 406 Shanghai Golden Bridge Bldg.
2077 Yan'an Road West, Hong Qiao District
Shanghai, PRC 200335
Tel: 86-21-6275-5700
Fax: 86 21-6275-5060

Singapore

Microchip Technology Taiwan
Singapore Branch
200 Middle Road
#10-03 Prime Centre
Singapore 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan, R.O.C

Microchip Technology Taiwan
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886 2-717-7175 Fax: 886-2-545-0139

EUROPE

United Kingdom

Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire SL8 5AJ
Tel: 44-1628-851077 Fax: 44-1628-850259

France

Arizona Microchip Technology SARL
Zone Industrielle de la Bonde
2 Rue du Buisson aux Fraises
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany

Arizona Microchip Technology GmbH
Gustav-Heinemann-Ring 125
D-81739 München, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy

Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-39-6899939 Fax: 39-39-6899883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa 222 Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

7/29/97

All rights reserved. ©1997, Microchip Technology Incorporated, USA. 8/97 Printed on recycled paper.

Information contained in this publication regarding device applications and the like is intended for suggestion only and may be superseded by updates. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.



Electromechanical Timer Replacement

Lawn Sprinkler System Using a PIC12C508

*Author: Aroldo Carvalho
Boca Raton, FL
USA
email: aroldo@gate.net*

APPLICATION OPERATION

This Lawn Sprinkler System is an electronic timer that can be programmed to activate and deactivate the sprinkler electric valve.

You can program it to activate anytime up to fifteen days, with 1 to 15 minutes for each sprinkler circuit. The sprinkler circuits will vary depending on the lawn installation. The number of sprinkler circuits are programmable.

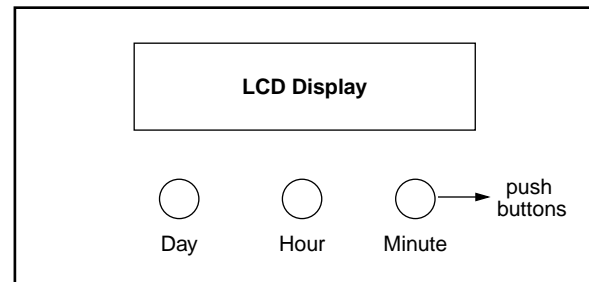
There will also be a bypass switch that will allow the user to activate the electric valve bypassing the electronic circuit at any time.

To improve the system a backup battery would hold the data in case of a power failure.

OPERATION

The user has an LCD Display and three push buttons to program the unit as shown in Figure 1.

FIGURE 1: SIMPLE DIAGRAM



1. Set the number of sprinkler circuits.

To set the number of sprinkler circuits, press and hold the day and hour buttons together, then press the minute button to select from 1 to 8 circuits.

2. Select the day, the start and end time for each sprinkler circuit.

Press the day button to scroll from 1 to 15, then press the hour and minute buttons to select the desired sprinkler circuit. Then enter start hour and minute and stop minute. The duration can be from 1 to 20 minutes per sprinkler circuit.

Repeat the above steps for each circuit.

Electromechanical Timer Replacement

THE SOFTWARE

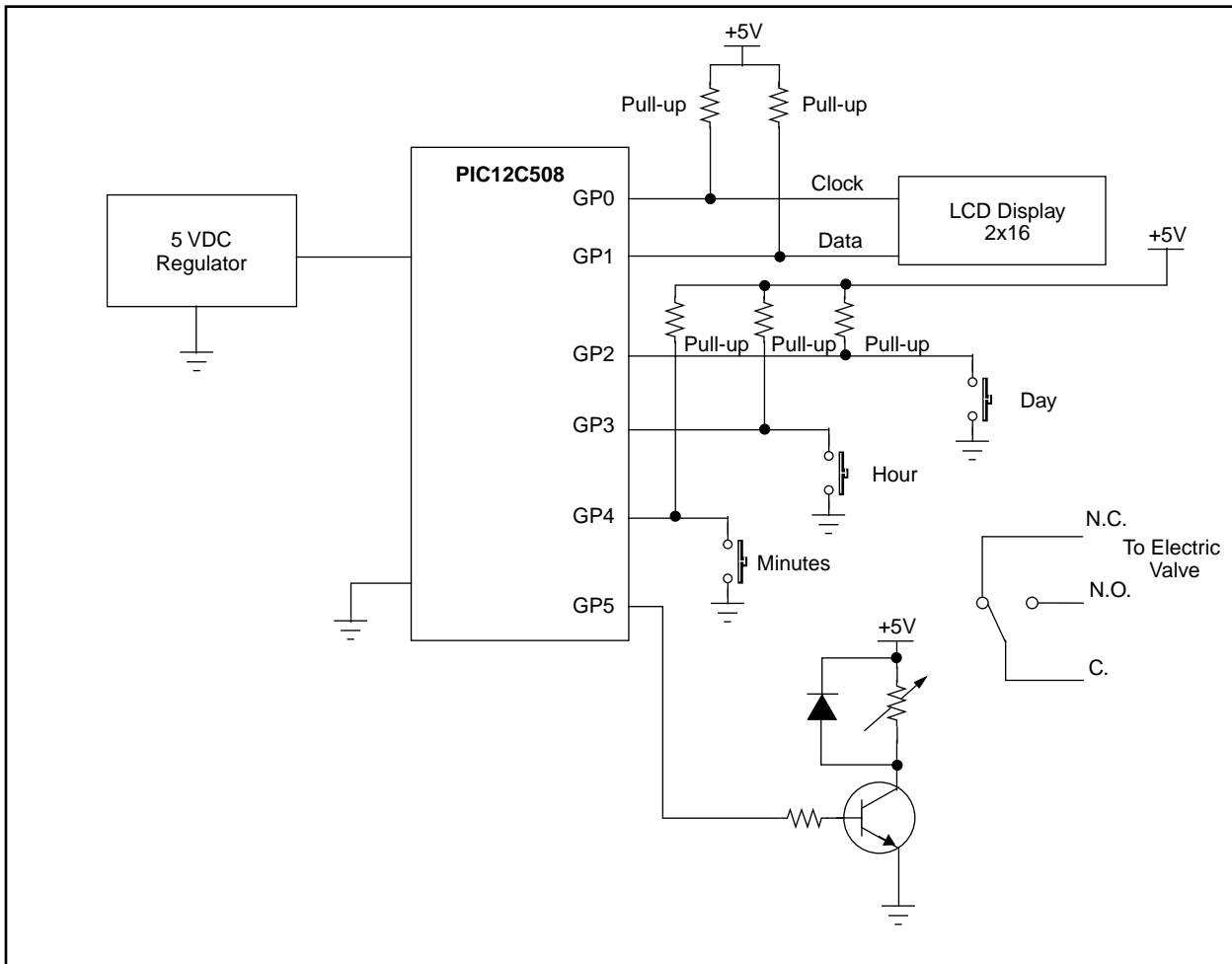
The software consists of a timer that will count seconds, minutes, hours and days up to fifteen days. After that it will start the first day again and so on. It then checks the activation/deactivation table and will turn on the relay, through GP5 output, for the amount of time programmed for each sprinkler circuit. The switching of the sprinkler system happens outside in the sprinkler gear using the water flow.

The pushbuttons are read by the software and will display the corresponding message depending on which button was pressed.

This is a low cost sprinkler system, it consists of the following components:

1. PIC12C508
2. 3 Push buttons
3. 6 Pull-up resistors (10 kOhms)
4. Serial LCD Display (2 Rows x 16 Columns)
5. NPN Transistor
6. Relay
7. 5 Volt Regulator
8. Wall Transformer (Input 110 VAC, output 9 VDC).
9. Diode
10. Bypass Switch.

FIGURE 2: SCHEMATIC





MICROCHIP

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office

Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602-786-7200 Fax: 602-786-7277
Technical Support: 602 786-7627
Web: <http://www.microchip.com>

Atlanta

Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston

Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508-480-9990 Fax: 508-480-8575

Chicago

Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas

Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 972-991-7177 Fax: 972-991-8588

Dayton

Microchip Technology Inc.
Two Prestige Place, Suite 150
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Los Angeles

Microchip Technology Inc.
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 714-263-1888 Fax: 714-263-1338

New York

Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516-273-5305 Fax: 516-273-5335

San Jose

Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto

Microchip Technology Inc.
5925 Airport Road, Suite 200
Mississauga, Ontario L4V 1W1, Canada
Tel: 905-405-6279 Fax: 905-405-6253

ASIA/PACIFIC

Hong Kong

Microchip Asia Pacific
RM 3801B, Tower Two
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2-401-1200 Fax: 852-2-401-3431

India

Microchip Technology Inc.
India Liaison Office
No. 6, Legacy, Convent Road
Bangalore 560 025, India
Tel: 91-80-229-4036 Fax: 91-80-559-9840

Korea

Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Shanghai

Microchip Technology
RM 406 Shanghai Golden Bridge Bldg.
2077 Yan'an Road West, Hong Qiao District
Shanghai, PRC 200335
Tel: 86-21-6275-5700
Fax: 86 21-6275-5060

Singapore

Microchip Technology Taiwan
Singapore Branch
200 Middle Road
#10-03 Prime Centre
Singapore 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan, R.O.C

Microchip Technology Taiwan
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886 2-717-7175 Fax: 886-2-545-0139

EUROPE

United Kingdom

Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire SL8 5AJ
Tel: 44-1628-851077 Fax: 44-1628-850259

France

Arizona Microchip Technology SARL
Zone Industrielle de la Bonde
2 Rue du Buisson aux Fraises
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany

Arizona Microchip Technology GmbH
Gustav-Heinemann-Ring 125
D-81739 München, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy

Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-39-6899939 Fax: 39-39-6899883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa 222 Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

7/29/97

All rights reserved. ©1997, Microchip Technology Incorporated, USA. 8/97  Printed on recycled paper.

Information contained in this publication regarding device applications and the like is intended for suggestion only and may be superseded by updates. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.



Electromechanical Timer Replacements

Maxi Functions, Micro Chip

*Author: Mark Walker
Circuit Graphics, Inc.
Hillsboro, OR
USA
email: mmwatcgi@aracnet.com*

This circuit allows fodder for several discussions, including efficient methods of using pins. A simple way to sense buttons with fewer pins than the classic cross-grid array does is one obvious derivative (5-lines for 8 push button, etc.). It also emphasizes the current driving capabilities of the Microchip family of parts in a fun environment.

DESCRIPTION

The application contains the kernel for several clever approaches for maximizing the use of the 8-pin family (even more can be done with A/D parts). It uses two 7-segment LED digits, a piezo noise maker and two push button to construct a simple game timer, and can be built with a minimum of components (could even be used like an alarm). This application shows how to drive 14 segments, a speaker and two push button with this 8-pin low-cost part.

To simplify understanding of the schematic, the displays are common anode type, and must have the driven cathode pulled low before the anode pin is pulled high to light any segment. The speaker is driven with all other pins low, and sleep is with all outputs low. The circuit is designed for 3 volts, and either push button should wake it from sleep.

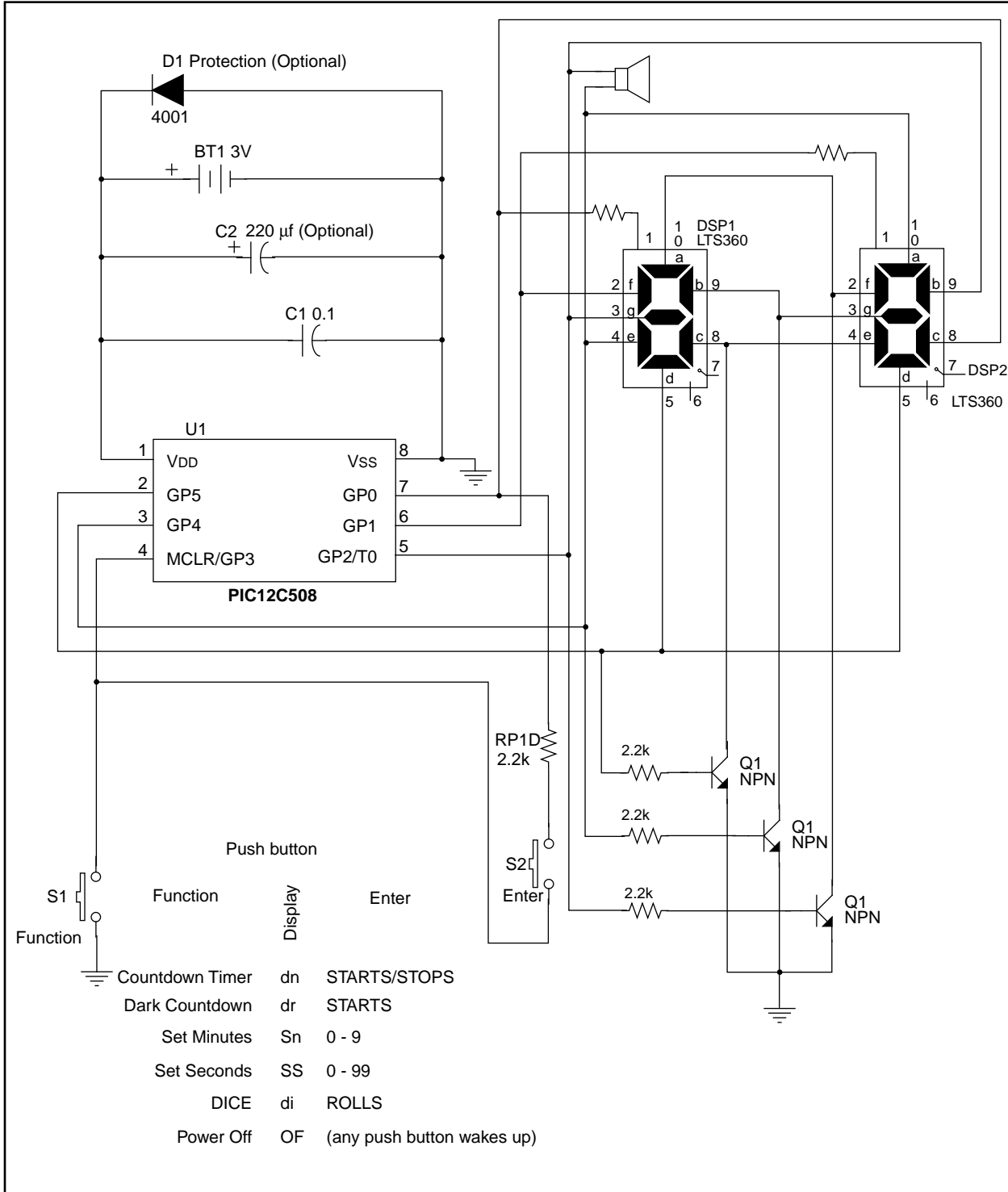
The functions mentioned on the schematic include a countdown timer, where the second button can start or stop the countdown, and a dark countdown timer, where the second button only starts the timer, with no visual hint as to when the time will run out (adds spice to some games). The minutes and seconds can be set and will be remembered between uses, until new values are entered. 9 minutes was selected as the maximum on that range because of the game nature of the circuit, and the less than perfect accuracy of the internal clock. 99 seconds was selected as a maximum on the set seconds function to maximize the functionality, so over 10 and a half minutes maximum is available. The dice mode is added for utility, and rolls new numbers every time the second button is pressed, (after a few beeps from the speaker) with first one digit popping up and then the other. The system goes to sleep after a bit of non-use and wakes up in the same place on any push. The OFF mode (OF) forces the unit to sleep with any button waking up and the unit starting at the top of the list.



Microchip Technology Incorporated, has been granted a non-exclusive, worldwide license to reproduce, publish and distribute all submitted materials, in either original or edited form. The author has affirmed that this work is an original, unpublished work and that he/she owns all rights to such work. All property rights, such as patents, copyrights and trademarks remain with author.

Electromechanical Timer Replacements

FIGURE 1: PIC12C508 GAME TIMER SCHEMATIC





MICROCHIP

WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office

Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602-786-7200 Fax: 602-786-7277
Technical Support: 602 786-7627
Web: <http://www.microchip.com>

Atlanta

Microchip Technology Inc.
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston

Microchip Technology Inc.
5 Mount Royal Avenue
Marlborough, MA 01752
Tel: 508-480-9990 Fax: 508-480-8575

Chicago

Microchip Technology Inc.
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas

Microchip Technology Inc.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 972-991-7177 Fax: 972-991-8588

Dayton

Microchip Technology Inc.
Two Prestige Place, Suite 150
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Los Angeles

Microchip Technology Inc.
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 714-263-1888 Fax: 714-263-1338

New York

Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516-273-5305 Fax: 516-273-5335

San Jose

Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto

Microchip Technology Inc.
5925 Airport Road, Suite 200
Mississauga, Ontario L4V 1W1, Canada
Tel: 905-405-6279 Fax: 905-405-6253

ASIA/PACIFIC

Hong Kong

Microchip Asia Pacific
RM 3801B, Tower Two
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2-401-1200 Fax: 852-2-401-3431

India

Microchip Technology Inc.
India Liaison Office
No. 6, Legacy, Convent Road
Bangalore 560 025, India
Tel: 91-80-229-4036 Fax: 91-80-559-9840

Korea

Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Shanghai

Microchip Technology
RM 406 Shanghai Golden Bridge Bldg.
2077 Yan'an Road West, Hong Qiao District
Shanghai, PRC 200335
Tel: 86-21-6275-5700
Fax: 86 21-6275-5060

Singapore

Microchip Technology Taiwan
Singapore Branch
200 Middle Road
#10-03 Prime Centre
Singapore 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan, R.O.C

Microchip Technology Taiwan
10F-1C 207
Tung Hua North Road
Taipei, Taiwan, ROC
Tel: 886 2-717-7175 Fax: 886-2-545-0139

EUROPE

United Kingdom

Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire SL8 5AJ
Tel: 44-1628-851077 Fax: 44-1628-850259

France

Arizona Microchip Technology SARL
Zone Industrielle de la Bonde
2 Rue du Buisson aux Fraises
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany

Arizona Microchip Technology GmbH
Gustav-Heinemann-Ring 125
D-81739 München, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy

Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-39-6899939 Fax: 39-39-6899883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa 222 Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

7/29/97

All rights reserved. ©1997, Microchip Technology Incorporated, USA. 8/97 Printed on recycled paper.

Information contained in this publication regarding device applications and the like is intended for suggestion only and may be superseded by updates. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.



Electromechanical Timer Replacement

PIC12C508 Based Timer

*Author: Ravi Pailoor
Chip Technologies
Bangalore, India*

DESCRIPTION

1. CONFIGURATION OF THE PIC12C508.
 - a) Pin 2 - input for configuration.
 - b) Pin 3 - output for software PWM generation.
 - c) Pin 4 - input for start/stop switch.
 - d) Pin 5 - input for 50 Hz time-base.
 - e) Pin 6 - output for relay driving.
 - f) Pin 7 - input for comparator input.
2. POWER SUPPLY.

Transformer T1, Diodes D1, D2, and D3 with C1, C2, C3, and U3 form the power supply giving 5 volts to the relay and the I.C.s. A transformerless power supply can be used if isolation is required.
3. TIME-BASE

To generate a time-base for the timer, the second opamp, U2B, is used to generate a square wave of 50 Hz. Alternatively, a resistor and a zener diode can be used for generating a near square wave. Even the internal clocking can be used for the time-base.
4. CONFIGURATION

Jumper J3 is used to select the range of the time-base. If J3 is open, 0 to 100 second range is selected and if closed 0 to 100 minutes range is selected.
5. START/STOP

Switch S1 will start the timing and also stop the timer is required.
6. COMPARISON

The PIC12C508 will generate PWM which is filtered to generate an analog signal. Double filtering can be used for a smoother waveform. This signal is fed to the inverting I/P of the opamp (LM358 used as the comparator). It is then compared to the signal at the non-inverting input. The signal (0 - 5V via potentiometer R5 and resistor R4) to the non-inverting input is proportional to the timing required.

7. OUTPUT

The SPDT relay is driven by the PIC12C508 on time-out.

OPERATION

On power up the timer goes to standby mode. The time is selected by R5. The range depends on the selection of J3 as explained previously. Once the time is set, the START/STOP button is pressed to start the timing. The PIC12C508 will generate a PWM signal at a ratio of 1:258 (8-bits) and poll pin 7 for a change of state. On detecting the change, the timing is scaled as follows:

256 bits = 100 seconds or minutes

$n \text{ bits} = (n \times 100)/256 \text{ seconds or minutes}$

After calculation the relay will be switched on and the timer will start timing out. The 50 Hz input is taken as the time-base and the timer will de-energize the relay on time-out and go to standby mode. At any given time, pressing switch S1 will stop the timer.

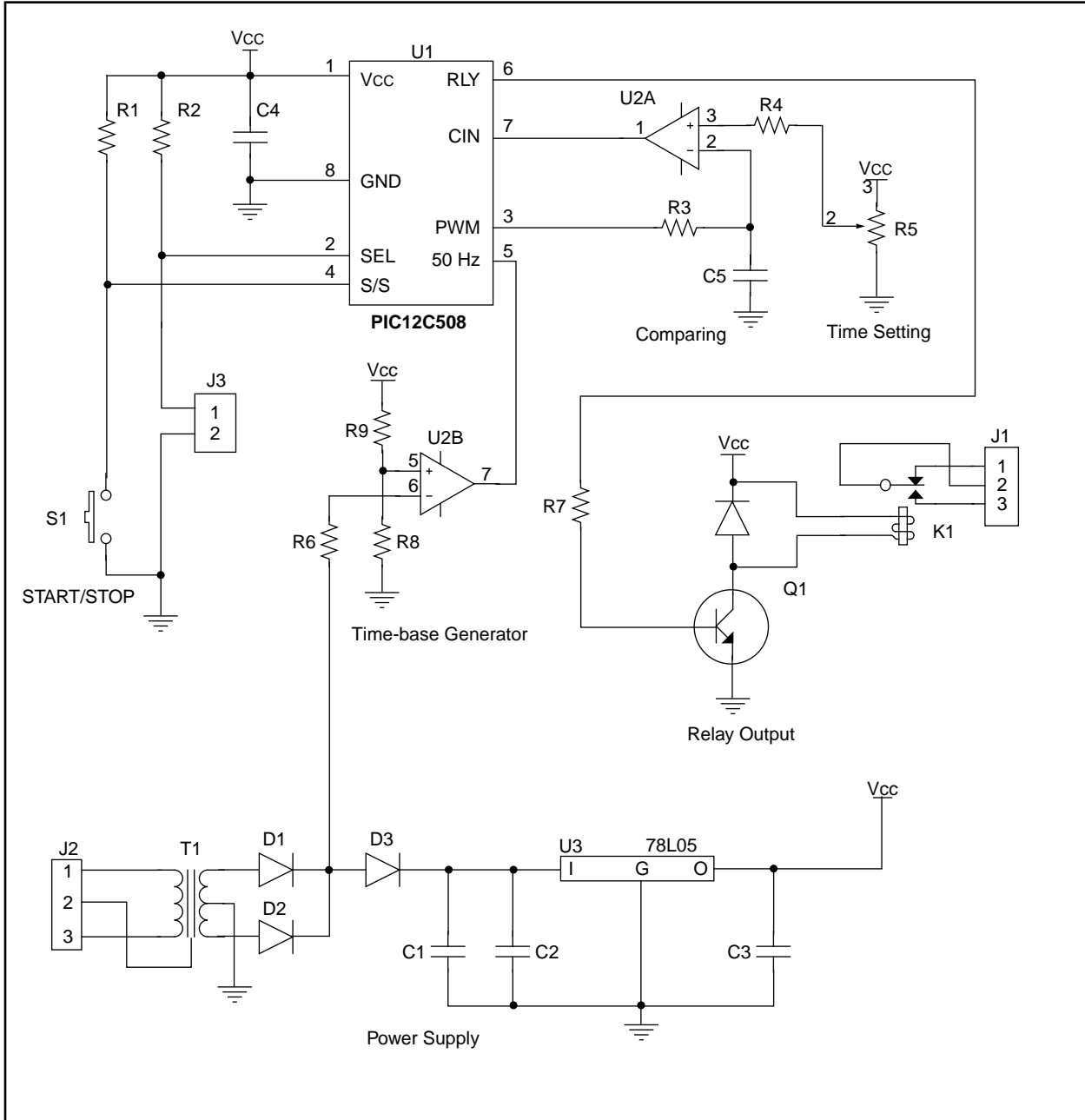
Notes:

1. Software generation of PWM and converting to analog will not give 5 volts due to attenuation. Hence the POT setting has to be limited to the generated voltage or the analog voltage will have to be amplified to 5 volts. This voltage will have to be proportional from 0 to 256 bits.
2. Internal or mains based timing is as accurate as crystal based timing.
3. Transformer based power supply is not cost effective but used mainly for isolation.
4. U2B for 50 Hz squaring is used because LM358 has dual opamp and only one is required for comparing.

Microchip Technology Incorporated, has been granted a non-exclusive, worldwide license to reproduce, publish and distribute all submitted materials, in either original or edited form. The author has affirmed that this work is an original, unpublished work and that he/she owns all rights to such work. All property rights, such as patents, copyrights and trademarks remain with author.

Electromechanical Timer Replacement

FIGURE 1: SCHEMATIC



Electromechanical Timer Replacement

A Simple Programmable Timer with Time Correction Circuit

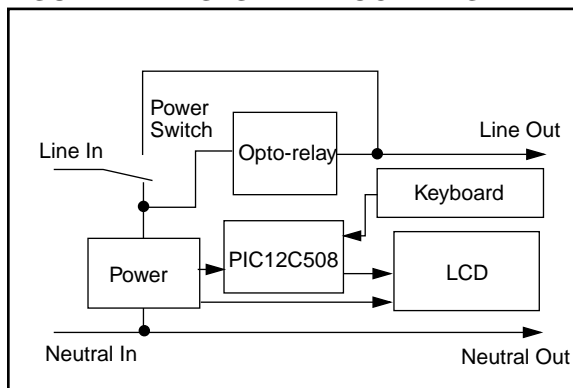
*Author: Kirill Yelizarov V.
Moscow Power Engineering
Institute
Moscow, Russia
email: tihonov@srv-*

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.

FIGURE 1: SYSTEM BLOCK DIAGRAM



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).

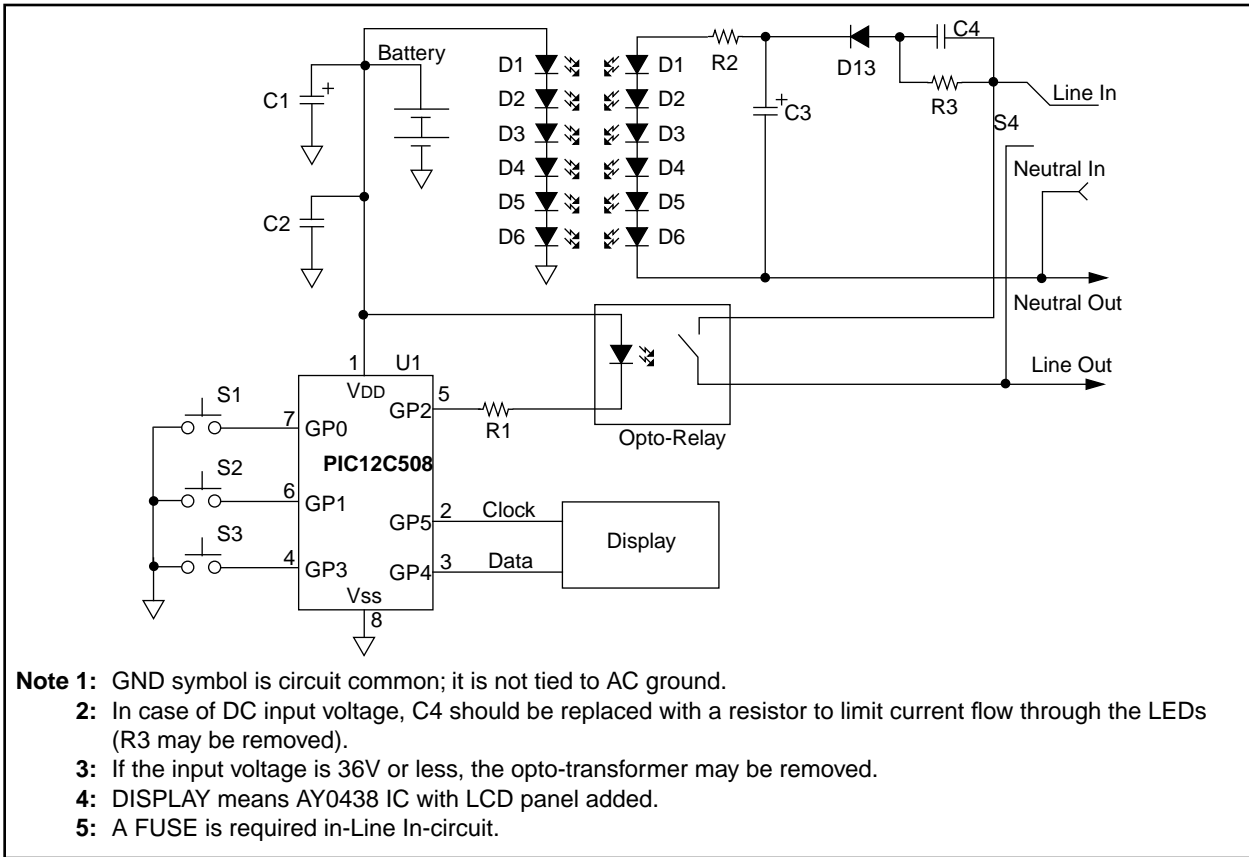
The opto-relay may be replaced with an opto-triac (AC operation only). Using a triac or an ordinary mechanical relay with an opto-relay or an opto-triac will power-up the output (DS00559). The opto-relay may be replaced with an ordinary mechanical relay, but most of them need a higher voltage and a transistor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

Electromechanical Timer Replacement

HARDWARE

The schematic diagram is shown in Figure 2.

FIGURE 2: SCHEMATIC DIAGRAM



The Timer Parts List

Capacitors:

- C1, C3 = 47 μ F electrolytic
- C2 = 0.1 μ F ceramic
- C4 = 1 μ F (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 = PICmicro™ output low voltage (0.6V max)
 V_{LED} = Input opto-relay IR LED voltage (0.8V typical)
 I_{LED} = LED current (10 mA typical)

$$R1 = \frac{3 - V_{LED} - V_O}{I_{LED}} = 160\Omega$$

- R2 = 100 Ω (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 M Ω , 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

Electromechanical Timer Replacement

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

;          ----- D A T A -----

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

;          ----- C O D E -----

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

;          ----- T A B L E S -----
;Decode data to 8 segment LCD
DecodeValue
          addwf       PCL,F
          retlw      b'00111111'         ;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

```
    retlw    b'01111111'    ;8
    retlw    b'01101111'    ;9
    retlw    b'00000000'    ;blank
    retlw    b'01000000'    ;hours/minutes separator
    retlw    b'01000000'    ;minutes/seconds separator
    retlw    b'00001000'    ;cursor

;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

    addwf    PCL,F

;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'
    retlw    b'11111111'
    retlw    b'11111111'

;
----- S U B R O U T I N E S -----
;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     TimerCount
    decfsz    TimePatch,F          ;This patch will increase accuracy
    return                                         ;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
    btfss    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
    goto     CorrectionSignal        ;skip for the normal operation

Main:
    btfsc    TimerStatus,StartStopFlag ;Change time if StartStopFlag is set
    call     GetTime
    btfss    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
    nop
    bcf      GPIO,LCD_Clock
    nop
    goto     CorrectionSignal
    org     0x1ff
    movlw    b'01110000'
    end
```



Electromechanical Timer Replacement

Reminder Timer for Changing Chemicals in a Water Softener (IRON)

*Author: Michael MacDonald
Mikaurie
Prescott, WI
USA
email: mikemd@presenter.com*

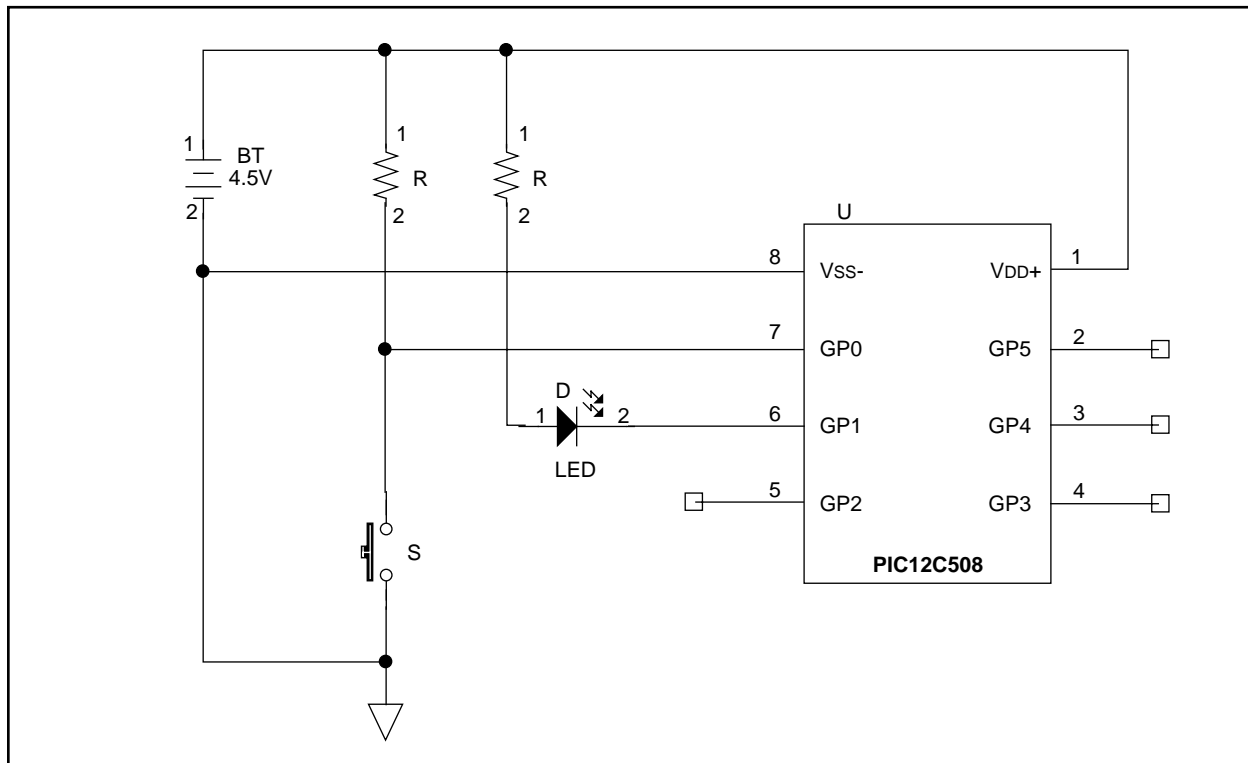


APPLICATION OPERATION:

When the battery is plugged in the unit flashes the LED for 1 second to verify good RAM and power up. A timing cycle then starts, that checks for approximately 180 days. At the end of the time period, the unit is placed in SLEEP mode. The LED flashes on for 100 ms every time it wakes up. Further time keeping is disabled until the battery is replaced. Then the cycle starts over.

This is an actual product that is being used by ECOWATER in Minnesota for reminding people to change the IRON cleaning chemicals in water softeners.

FIGURE 1: SCHEMATIC



Microchip Technology Incorporated, has been granted a non-exclusive, worldwide license to reproduce, publish and distribute all submitted materials, in either original or edited form. The author has affirmed that this work is an original, unpublished work and that he/she owns all rights to such work. All property rights, such as patents, copyrights and trademarks remain with author.

Electromechanical Timer Replacement

APPENDIX A: SOURCE CODE

```
PICMaster with 12C5XX probe.
Assembler/Compiler version:
Current version shipping with MPLAB 3.22
Include files:
; P12C508.INC Standard Header File, Version 1.01 Microchip Technology, Inc.
NOLIST
; This header file defines configurations, registers, and other useful bits of
; information for the PIC12C508 microcontroller.
; These names are taken to match the data sheets as closely as possible.
; Note that the processor must be selected before this file is included.
; The processor may be selected the following ways:
;
; 1. Command line switch:
;    C:\ MPASM MYFILE.ASM /P12C508
;
; 2. LIST directive in the source file
;    LIST P=12C508
;
; 3. Processor Type entry in the MPASM full-screen interface

;=====
;
; Revision History
;
;=====

;Rev:   Date:   Reason:

;1.01   08/21/96 Removed VCLMP fuse, corrected oscillators
;1.00   04/10/96 Initial Release

;=====
;
; Verify Processor
;
;=====

        IFNDEF __12C508
            MESSG "Processor-header file mismatch. Verify selected processor."
        ENDIF

;=====
;
; Register Definitions
;
;=====

        W      EQU    H'0000'
        F      EQU    H'0001'

;----- Register Files -----

        INDF   EQU    H'0000'
        TMR0   EQU    H'0001'
        PCL    EQU    H'0002'
        STATUS EQU    H'0003'
        FSR    EQU    H'0004'
        OSCCAL EQU    H'0005'
        GPIO   EQU    H'0006'

#define     GPWUF   STATUS,7

;----- PORT BIT ASSIGNMENTS -----
GPIO_DIR   EQU    0X019

#define     S1      GPIO,0      ;IN
```

Electromechanical Timer Replacement

```
#DEFINE LED GPIO,1 ;OUT
#DEFINE TOGGLE GPIO,2 ;OUT
#DEFINE NC GPIO,3 ;IN

;----- STATUS Bits -----

#DEFINE PA2 STATUS,7
#DEFINE PA1 STATUS,6
#DEFINE PA0 STATUS,5
#DEFINE TO STATUS,4
#DEFINE PD STATUS,3
#DEFINE Z STATUS,2
#DEFINE DC STATUS,1
#DEFINE C STATUS,0

;----- OPTION Bits -----

T0CS EQU H'0005'
T0SE EQU H'0004'
PSA EQU H'0003'
PS2 EQU H'0002'
PS1 EQU H'0001'
PS0 EQU H'0000'

;=====
;
; RAM Definition
;
;=====

__MAXRAM H'1F'

;=====
;
; Configuration Bits
;
;=====

_MCLRE_ON EQU H'0FFF'
_MCLRE_OFF EQU H'0FEF'
_CP_ON EQU H'0FF7'
_CP_OFF EQU H'0FFF'
_WDT_ON EQU H'0FFF'
_WDT_OFF EQU H'0FFB'
_LP_OSC EQU H'0FFC'
_XT_OSC EQU H'0FFD'
_IntRC_OSC EQU H'0FFE'
_ExtRC_OSC EQU H'0FFF'

;*****
;* RAM AREA
;*****

MS_COUNTER EQU 0X07
SEC_COUNTER EQU 0X08
MIN_COUNTER EQU 0X09
HOUR_COUNTER EQU 0X0A
DAYS_COUNTER EQU 0X0B

COUNT_OFF EQU 0X0C
BITS_COUNT EQU 0X0D
XMIT_COUNT EQU 0X0E
BUTTON_MEMORY EQU 0X0F

COUNT1 EQU 0X010
COUNT2 EQU 0X011
```

Electromechanical Timer Replacement

```
COUNT3      EQU      0X012

;*****
;  MACROS
;*****

LED_ON      MACRO
            BCF      LED
            ENDM

LED_OFF     MACRO
            BSF      LED
            ENDM

;Software listing:

LIST       P=12C509, W=2

            #INCLUDE "P12C508.INC"

            __FUSES  _CP  _OFF&_WDT_ON&_LP_OSC&_MCLRE_ON

;  NOTE: CHANGE _CP_OFF TO _CP_ON TO CODE PROTECT PARTS DURING PRODUCTION CYCLE.

;PAC SYSTEM TRANSMITTER

;

;-----START OF PROGRAM MEMORY AREA-----

            ORG      0          ;RESET AREA
            GOTO    START

;*****
;  SET UP THE I/O CHANNELS
;*****

SET_IO     MOVLW   0X08F      ;DISABLE WAKE UP ON I/O CHANGE
            ;AND ENABLE WEAK PULL UPS
            OPTION   ;
            MOVLW   GPIO_DIR  ;SET UP I/O PORTS
            TRIS    6
            RETURN

;*****
;  TURN ON LED - TOTAL TIME 1 SEC
;*****

FLASH_LED  LED_ON          ;TURN ON THE LED
            MOVLW   .250     ;PRESET TIMER VALUE
            MOVWF   COUNT1
            MOVLW   .8
            MOVWF   COUNT2

FLASH_LOOP CLRWDT          ;RESET THE WDT
            DECFSZ  COUNT1
            GOTO    FLASH_LOOP
            DECFSZ  COUNT2
            GOTO    FLASH_LOOP
            LED_OFF          ;TURN OFF THE LED
            RETURN

;*****
;  TURN ON LED - TOTAL TIME 100 MS
;*****
```

Electromechanical Timer Replacement

```
FLASH_LED_S    LED_ON          ;TURN ON THE LED
               MOVWLW    .205    ;PRESET TIMER VALUE
               MOVWF    COUNT1
               MOVWLW    .1
               MOVWF    COUNT2
FLASH_LOOP_S   CLRWDT          ;RESET THE WDT
               DECFSZ   COUNT1
               GOTO    FLASH_LOOP_S
               DECFSZ   COUNT2
               GOTO    FLASH_LOOP_S
               LED_OFF    ;TURN OFF THE LED
               RETURN

;*****
;          SLEEP MODE
;*****

SLEEP_MODE    MOVWLW    0X08F    ;DISABLE WAKE UP ON I/O CHANGE
               ;AND ENABLE WEAK PULL UPS
               OPTION    ;
               OPTION    ;NEEDED TO INSURE OPTION REG. LOADED
               SLEEP

;*****
;          INITIALIZATION ROUTINE
;*****

START         MOVWF    STATUS    ;FIND OUT HOW WE GOT HERE
               ANDLW    0X018    ;MASK OFF TO AND PD
               XORLW    0X00    ;WAS IT WAKE UP FROM SLEEP?
               BTFSC   Z
               GOTO    FLASH_TIME ;YES! FLASH LED
               MOVWF    STATUS    ;FIND OUT HOW WE GOT HERE
               ANDLW    0X018    ;MASK OFF TO AND PD
               XORLW    0X08    ;DID WDT TIME OUT?
               BTFSS   Z
               GOTO    CLEAR_RAM
FLASH_TIME    CALL    SET_IO    ;SET UP I/O PINS
               CALL    FLASH_LED_S
               GOTO    SLEEP_MODE

;-----CLEAR ALL RAM LOCATIONS-----

CLEAR_RAM     MOVWLW    0X08F    ;DISABLE WAKE UP ON I/O CHANGE
               ;AND ENABLE WEAK PULL UPS
               OPTION    ;
               CALL    SET_IO    ;CHECK FOR TEST MODE
               MOVWLW    0X07    ;PUT '7' IN THE FSR
               MOVWF    FSR    ;TO INITIALIZE ALL RAM LOCATIONS
LP           CLRW
               CLRF    INDF    ;CLEAR THE INDF REGISTER
               INCF    FSR,F    ;INCREMENT THE ADDRESS POINTER
               MOVWLW    0X0E0    ;CHECK FOR LAST BYTE
               XORWF    FSR,W
               BTFSS   Z
               GOTO    LP
               CALL    FLASH_LED

;-----TEST RAM FOR VALID DATA-----

               CALL    RAM_TEST    ;CHECK THE RAM THAT WILL BE USED
               BTFSC   S1          ;IS GP0 GROUNDED?
               GOTO    MAIN
               MOVWLW    .182      ;PRESET THE DAYS COUNTER
               MOVWF    DAYS_COUNTER
```

Electromechanical Timer Replacement

```
        MOVLW      .23          ;PRESET THE HOUR COUNTER
        MOVWF     HOUR_COUNTER
        MOVLW     .59          ;PRESET THE MINUTES COUNTER
        MOVWF     MIN_COUNTER
        MOVLW     .45          ;PRESET THE SECONDS COUNTER
        MOVWF     SEC_COUNTER

;*****
;      MAIN PROGRAM LOOP
;*****

MAIN      NOP
          NOP
          CALL     SET_IO      ;SET UP THE DATA

;-----THIS IS THE MAIN TIMING LOOP SECTION-----

          MOVLW     .200       ;SET TIMING LOOP (HARDWARE)
          MOVWF     COUNT1
MAIN_LOOP CLRWDI
          DECFSZ    COUNT1
          GOTO     MAIN_LOOP
          NOP

;-----THIS IS THE TIMING SECTION FOR COUNTING TO 180 DAYS-----

          INCF     MS_COUNTER   ;CHECK FOR VALUE
          MOVLW     .10         ;CHECK 10 COUNTS
          XORWF    MS_COUNTER,W ;MATCH YET?
          BTFSS    Z
          GOTO     MAIN        ;NO!

;-----TOGGLE OUTPUT PIN-----

          BTFSS    TOGGLE      ;CHECK CURRENT STATE OF PIN
          GOTO     TOGGLE_ON
          BCF     TOGGLE      ;TURN PIN OFF (LOW)
          GOTO     CHECK_TIME
TOGGLE_ON BSF     TOGGLE      ;TURN PIN ON (HIGH)
          NOP

CHECK_TIME CLRF     MS_COUNTER
          INCF     SEC_COUNTER  ;CHECK FOR 60 SECONDS
          MOVLW     .60
          XORWF    SEC_COUNTER,W ;MATCH YET?
          BTFSS    Z
          GOTO     MAIN        ;NO!
          CLRF     SEC_COUNTER
          INCF     MIN_COUNTER  ;CHECK FOR 60 MINUTES
          MOVLW     .60
          XORWF    MIN_COUNTER,W ;MATCH YET?
          BTFSS    Z
          GOTO     MAIN        ;NO!
          CLRF     MIN_COUNTER
          INCF     HOUR_COUNTER ;CHECK FOR 24 HOURS
          MOVLW     .24
          XORWF    HOUR_COUNTER,W ;MATCH YET?
          BTFSS    Z
          GOTO     MAIN        ;NO!
          CLRF     HOUR_COUNTER
          INCF     DAYS_COUNTER ;CHECK FOR 180 DAYS
          MOVLW     .183
          XORWF    DAYS_COUNTER,W
          BTFSS    Z
          GOTO     MAIN        ;NO!
          CALL     FLASH_LED_S ;FLASH THE LED
```



Electromechanical Timer Replacement

Solutions Cubed Real-Time Clock

*Author: David Brobst
Solutions Cubed
Chico, CA
USA
email: solcubed@solutions*

OVERVIEW

This design fragment is based upon converting an electromechanical timer idea to a PIC12CXXX 8-bit microcontroller.

DESIGN IDEA

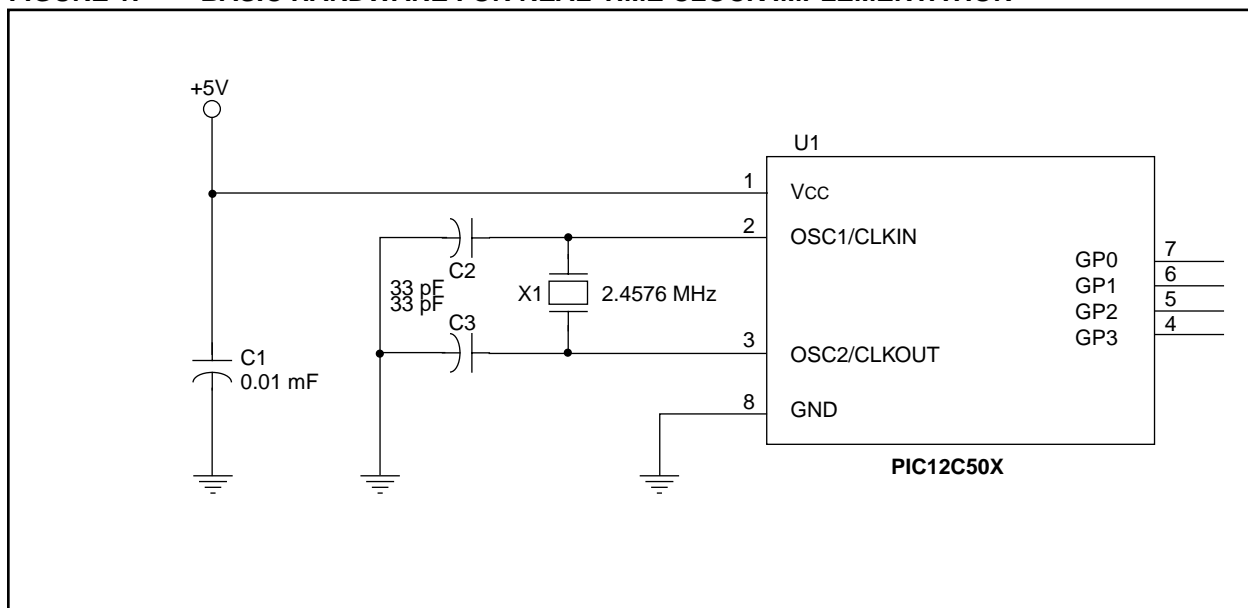
This design idea uses the PIC12C50X series of 8-pin microcontrollers to implement a medium accuracy real-time clock. There are two unique features to this design when compared to other real-time clock designs. The first is that common asynchronous communication baud rates can be easily implemented. The other is that leap year compensation is implemented in a straight forward and simple manner. Figure 1 shows the basic hardware for the design.

HARDWARE METHODOLOGY

The heart of the system is the 2.4576MHz (X1) crystal which can be found from any of the leading crystal manufacturers. The neat thing about this value is that it allows for an easy clock breakdown for asynchronous communication and allows for a fairly easy implementation of a real-time clock. As with any real-time clock, the accuracy of the crystal and the value of its load are the major factors in determining clock accuracy. In this case, X1 can be easily obtained with a 20 ppm accuracy and a 16 pF load.

By using the internal MCLR of the PIC12CXXX family, an extra input pin is made available. C1 is used for decoupling purposes.

FIGURE 1: BASIC HARDWARE FOR REAL-TIME CLOCK IMPLEMENTATION



Microchip Technology Incorporated, has been granted a non-exclusive, worldwide license to reproduce, publish and distribute all submitted materials, in either original or edited form. The author has affirmed that this work is an original, unpublished work and that he/she owns all rights to such work. All property rights, such as patents, copyrights and trademarks remain with author.

Electromechanical Timer Replacement

SOFTWARE METHODOLOGY

Appendix A gives the code listing which will be discussed here.

As with almost all clock designs, a simple counter is used to keep track of the time. With a 2.4576 MHz crystal, the internal instruction cycle is 1.627604 μ s, which at first glance does not seem very promising. However, there are exactly 61440 instruction cycles per 100 ms using this clock frequency. Using TMR0 with a 256 prescaler this breaks down to an overflow every 240 counts. Therefore TMR0 is preloaded with d'11' every 100 ms so that TMR0 will overflow in exactly 100 ms.

The code knows that TMR0 has overflowed with a simple compare register TMR_OLD. If TMR0 is less than TMR_OLD then the timer has rolled over and it is time to update the real-time clock. After ten rollovers, the SECONDS register is incremented and so forth through the rest of the code.

In order to take into account the lag, when TMR0 starts counting after a write, and the prescaler being erased after a write, a timing loop is employed that implements an average error wait every time through the loop. This is the major source of error in the timekeeping process. The delay loop could be tailored to meet the individual cases of the code that the clock was implemented in, especially if the system was deterministic.

In order to not miss a rollover, the routine must be checked within 100 ms of the previous roll over.

This code keeps track all the way through years, with leap year compensation. The MONTH_TABLE routine is a simple computed GOTO look up table, with a special circumstance for February. Leap year occurs every four years, with the added bonus that the years it occurs on can be evenly divided by four. This means that if the YEARS register's two least significant bits are zeros it is a leap year.

The last important bit of coding is that the YEARS register is merely a count up register, so that the year 2000 could be represented by d'100', while 1900 would be d'00'. This is to help code get over the year 2000 hump. Before this type of counting would be a problem, it will be the year 2156. Hopefully, code and devices implemented now will not still be in service.

The code of interest is in the subroutine RTC. RTC calls MONTH_TABLE. This means that the PIC12C50X'S limited stack would be used up if RTC was used as a subroutine from the main program loop. However, it is relatively simple to put RTC into a straight line code, along with MONTH_TABLE. This way the whole thing could be in the main program loop and not impact precious stack depth, or subroutine space. It is presented in this manner to ease readability and understanding.

Further Expansion

Because of the clock frequency, common baud rates (2400, 9600, 19200) are easily obtainable and do not have the error associated with using off value clocks. Also, the speed of the clock allows for some fairly rigorous computational efforts to be realized along with an on-board time stamp.

Comparisons

This real-time clock is a good fit for applications where a moderate accuracy time, along with communication, is necessary while still meeting a low price and parts count. A 32.768 kHz solution is a better fit where accuracy is important (because of the prescaler and offset problems with the 2.4576 MHz version), or where low power consumption is vital.

Current Use

The ideas presented here have been incorporated into a product currently being offered by Solutions Cubed. It includes an alarm output along with serial communication capabilities.

RAM Used:8 bytes, 1 byte is a TEMP register

Subroutine Bytes:79

Program Bytes (as presented):110

Program Cycles (min, no roll over):9

Program Cycles (max, everything changes): 557

MICROCHIP TOOLS USED

Assembler/Compiler Version

MPLAB 3.22.02 and MPASM 1.50

Electromechanical Timer Replacement

APPENDIX A: SOURCE CODE

Appendix A: Code Listing

MPASM 01.50 Released MICROCLK.ASM 5-29-1997 13:01:23 PAGE 1

```
LOC OBJECT CODE      LINE SOURCE TEXT
VALUE

00001 ;*****
00002 ;*****
00003 ;****              SOLUTIONS CUBED              ****
00004 ;****              Frank Rossini, Lon Glazner, David Brobst      ****
00005 ;*****
00006 ;*****
00007 ;
00008 ;
00009 ;*****
00010 ;****              Solutions Cubed Real Time Clock      ****
00011 ;*****
00012 ;
00013 ;          The purpose of this code is to develop a real time clock which
00014 ;can interface directly and easily to a standard asynchronous communications
00015 ;channel using the PIC12C50X chip.
00016 ;
00017 ;*****
00018 ;
00019 ;
00020 ;*****
00021 ;*****
00022 ;****  Define registers, constants, processor, and assembler directives  ****
00023 ;*****
00024 ;*****
00025 ;
00026 ;Processor
00027 ;
00028          LIST      P=12C508              ;Processor used
00029 ;
00030 ;Processor defined registers and bits
00031 ;
00032          INCLUDE "C:\PIC\HEADERS\P12C508.INC"      ;Microchip include file
00001          LIST
00002 ; P12C508.INC Standard Header File, Version 1.02      Microchip Technology, Inc.
00105          LIST
00033 ;
00034 ;Program defined registers
00035 ;
00000007 00036 TEMP0          EQU      H'07'              ;Pseudo-WORKING registers
00037 ;
00000008 00038 TMR_OLD          EQU      H'08'
00000009 00039 BIN1            EQU      H'09'              ;Time keeping registers
0000000A 00040 SECONDS          EQU      H'0A'
0000000B 00041 MINUTES          EQU      H'0B'
0000000C 00042 HOURS            EQU      H'0C'
0000000D 00043 DAYS            EQU      H'0D'
0000000E 00044 MONTHS          EQU      H'0E'
0000000F 00045 YEARS            EQU      H'0F'
00046 ;
00047 ;*****
00048 ;
00049 ;
00050 ;*****
00051 ;*****
00052 ;****              Reset Vector              ****
00053 ;*****
```

Electromechanical Timer Replacement

```
00054 ;*****
0000 00055     ORG     H'000'
0000 0A50 00056     GOTO    MAIN
00057 ;*****
00058 ;
00059 ;
00060 ;*****
00061 ;*****
00062 ;***                Time Routines                ***
00063 ;*****
00064 ;*****
00065 ;MONTH_TABLE -- Keeps track of number of days per month
00066 ;RTC -- Routine for real time clock
00067 ;*****
00068 ;
00069 ;
00070 ;*****
00071 ;MONTH_TABLE: This table keeps track of the number of days in each month.
00072 ;It is not adjusted for leap year. The NOP in the beginning of the table is
00073 ;because the first month, January, is denoted 1. The MONTHS registers is
00074 ;assumed to be pre-loaded into W before this routine is called.
00075 ;    Called From: TIME_INCREMENT
00076 ;    Modified Registers: PCL, STATUS, TEMPO
00077 ;    Subroutines Called: NONE
00078 ;
0001 00079 MONTH_TABLE
0001 01E2 00080     ADDWF   PCL,F
0002 0000 00081     NOP
0003 0820 00082     RETLW   H'20'                ;d31 --- # of days in January
0004 0A0F 00083     GOTO    CHECK_FEB           ;Leap year compensation
0005 0820 00084     RETLW   H'20'                ;d31 --- # of days in March
0006 081F 00085     RETLW   H'1F'                ;d30 --- # of days in April
0007 0820 00086     RETLW   H'20'                ;d31 --- # of days in May
0008 081F 00087     RETLW   H'1F'                ;d30 --- # of days in June
0009 0820 00088     RETLW   H'20'                ;d31 --- # of days in July
000A 0820 00089     RETLW   H'20'                ;d31 --- # of days in August
000B 081F 00090     RETLW   H'1F'                ;d30 --- # of days in September
000C 0820 00091     RETLW   H'20'                ;d31 --- # of days in October
000D 081F 00092     RETLW   H'1F'                ;d30 --- # of days in November
000E 0820 00093     RETLW   H'20'                ;d31 --- # of days in December
000F 00094 CHECK_FEB
000F 020F 00095     MOVF    YEARS,W                ;Leap years are divisible by 4
0010 0027 00096     MOVWF   TEMPO                ;     therefore, two RRF should
0011 060F 00097     BTFSC   YEARS,0                ;     in the C bit
0012 081D 00098     RETLW   H'1D'                ;d28 --- Regular February
0013 062F 00099     BTFSC   YEARS,1
0014 081D 00100     RETLW   H'1D'                ;d28 --- Regular February
0015 081E 00101     RETLW   H'1E'                ;d29 --- Leap year
00102 ;*****
00103 ;
00104 ;
00105 ;*****
00106 ;RTC: This routine is used keep track of the real time of the program.
00107 ;    Called From: MAIN_LOOP
00108 ;    Registers Used: BIN1, DAYS, HOURS, MINUTES, MONTHS, SECONDS,
00109 ;                   STATUS, TEMPO, TMR_OLD, TMR0, YEARS
00110 ;    Subroutines Called: MONTH_TABLE
00111 ;
0016 00112 RTC
0016 0208 00113     MOVF    TMR_OLD,W                ;Check to see if TMR0 rolled over
0017 0081 00114     SUBWF   TMR0,W                ;     during MORE_PROGRAM
0018 0603 00115     BTFSC   STATUS,C                ;If C set then no roll over
0019 0A4F 00116     GOTO    RTC_END
001A 00117 TMR0_OFFSET
001A 0201 00118     MOVF    TMR0,W                ;Get offset correct
001B 0028 00119     MOVWF   TMR_OLD
```

Electromechanical Timer Replacement

```
001C 0201 00120 T00_0  MOVF    TMR0,W           ;Make sure TMR0 has incremented
001D 0088 00121      SUBWF   TMR_OLD,W
001E 0643 00122      BTFSC  STATUS,Z           ;If not equal then TMR0 has increment
001F 0A1C 00123      GOTO   T00_0
0020 0C52 00124      MOVLW  H'52'           ;Equalize TMR0 prescale error
0021 0027 00125      MOVWF  TEMPO
0022 0000 00126      NOP
0023 02E7 00127 T00_1  DECFSZ  TEMPO,F
0024 0A23 00128      GOTO   T00_1
0025 0C11 00129      MOVLW  H'11'           ;Put in offset
0026 01E1 00130      ADDWF  TMR0,F
0027 0201 00131      MOVF   TMR0,W           ;Re-load so don't miss roll over
0028 0028 00132      MOVWF  TMR_OLD
0029      00133 TIME_INCREMENT
0029 02E9 00134      DECFSZ  BIN1,F           ;See if has been 1 second
002A 0A4F 00135      GOTO   TI_END
002B 02AA 00136      INCF   SECONDS,F       ;Increment SECONDS
002C 0C3C 00137      MOVLW  H'3C'           ;See if MINUTES should be incremented
002D 008A 00138      SUBWF  SECONDS,W
002E 0743 00139      BTFSS  STATUS,Z           ;If Z set then increment MINUTES
002F 0A4D 00140      GOTO   TI_RESET
0030 006A 00141      CLRF  SECONDS           ;Reset SECONDS
0031 02AB 00142      INCF  MINUTES,F        ;Increment MINUTES
0032 0C3C 00143      MOVLW  H'3C'           ;See if HOURS should be incremented
0033 008B 00144      SUBWF  MINUTES,W
0034 0743 00145      BTFSS  STATUS,Z           ;If Z set then increment HOURS
0035 0A4D 00146      GOTO   TI_RESET
0036 006B 00147      CLRF  MINUTES           ;Reset MINUTES
0037 02AC 00148      INCF  HOURS,F          ;Increment HOURS
0038 0C18 00149      MOVLW  H'18'           ;See if DAYS should be incremented
0039 008C 00150      SUBWF  HOURS,W
003A 0743 00151      BTFSS  STATUS,Z           ;If Z set then increment DAYS
003B 0A4D 00152      GOTO   TI_RESET
003C 006C 00153      CLRF  HOURS            ;Reset HOURS
003D 02AD 00154      INCF  DAYS,F           ;Increment Days
003E 020E 00155      MOVF   MONTHS,W
003F 0901 00156      CALL  MONTH_TABLE      ;Get number of days in month
0040 008D 00157      SUBWF  DAYS,W
0041 0743 00158      BTFSS  STATUS,Z           ;If Z set then month over
0042 0A4D 00159      GOTO   TI_RESET
0043 0C01 00160      MOVLW  H'01'           ;Reset DAYS
0044 002D 00161      MOVWF  DAYS
0045 02AE 00162      INCF  MONTHS,F        ;Increment MONTHS
0046 0C0D 00163      MOVLW  H'0D'           ;See if at end of year
0047 008E 00164      SUBWF  MONTHS,W
0048 0743 00165      BTFSS  STATUS,Z           ;If Z set then at end of year
0049 0A4D 00166      GOTO   TI_RESET
004A 0C01 00167      MOVLW  H'01'           ;Reset MONTHS
004B 002E 00168      MOVWF  MONTHS
004C 02AF 00169      INCF  YEARS,F
004D      00170 TI_RESET
004D 0C0A 00171      MOVLW  H'0A'           ;Reset the number of times for 100ms
004E 0029 00172      MOVWF  BIN1           ; overflow
004F      00173 TI_END
004F 0800 00174 RTC_END RETLW  H'00'
00175 ;*****
00176 ;
00177 ;
00178 ;*****
00179 ;*****
00180 ;*****
00181 ;****                               Main Program                               ****
00182 ;*****
00183 ;*****
00184 ;*****
00185 ;
```

Electromechanical Timer Replacement

```
00186 ;
00187 ;*****
0050 00188 MAIN
00189 ;
0050 00190 CLEAR_REGISTERS
0050 0067 00191 CLR FSR ;Clear first RAM location for use
0051 0C18 00192 MOVLW H'18' ;Number of registers to clear
0052 0027 00193 MOVWF TEMP0
0053 0C08 00194 MOVLW H'08' ;Start of RAM clearing
0054 0024 00195 MOVWF FSR
0055 00196 CLEAR_LOOP
0055 0060 00197 CLR INDF ;Clear register pointed to
0056 02A4 00198 INC FSR,F ;Go to next RAM location to clear
0057 02E7 00199 DECFSZ TEMP0,F ;Check to see if all clearing done
0058 0A55 00200 GOTO CLEAR_LOOP
0059 00201 PORT_SETUP
0059 0C3B 00202 MOVLW H'3B' ;0011 1011
005A 0026 00203 MOVWF GPIO
005B 0C3B 00204 MOVLW H'3B' ;0011 1011
005C 0006 00205 TRIS GPIO
005D 00206 OPTION_SETUP
005D 0CC7 00207 MOVLW H'C7' ;1100 0111 -- Wake up disabled, weak
005E 0002 00208 OPTION ; PUs disabled, internal TMR0,
005F 00209 TIME_SETUP ; 1:256 prescaler to TMR0
005F 006A 00210 CLR SECONDS ;Set a beginning time: 12:00AM,
0060 006B 00211 CLR MINUTES ; January, 1 1996
0061 006C 00212 CLR HOURS
0062 0C01 00213 MOVLW H'01'
0063 002D 00214 MOVWF DAYS
0064 002E 00215 MOVWF MONTHS
0065 0C60 00216 MOVLW H'60' ;d96
0066 002F 00217 MOVWF YEARS
0067 0C0A 00218 MOVLW H'0A' ;Overflow for 100mS register
0068 0029 00219 MOVWF BIN1
0069 0C11 00220 MOVLW H'11' ;Set up for 100mS overflow
006A 0021 00221 MOVWF TMR0 ;Set up for first find
006B 0028 00222 MOVWF TMR_OLD
006C 00223 MAIN_LOOP
006C 0916 00224 CALL RTC ;Time Routines
006D 0A6C 00225 GOTO MAIN_LOOP
00226 ;*****
00227 ;
00228 ;End of code indicator
00229 ;
00230 END
```

MEMORY USAGE MAP ('X' = Used, '-' = Unused)

```
0000 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX
0040 : XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXX-- -----
```

All other memory blocks unused.

Program Memory Words Used: 110
Program Memory Words Free: 402

Errors : 0
Warnings : 0 reported, 0 suppressed
Messages : 0 reported, 0 suppressed



Electromechanical Timer Replacement

The Galactic Timer

*Author: Michael Kirkhart
GSE, Inc.
Farmington Hills, MI
USA
email: kirkhart@rust.net*



OVERVIEW

The Swiss Army Timer is a generic, programmable timer circuit which can electronically replicate the functionality of some of the more popular electromechanical timer relay circuits. Using a PIC12C508 along with a 24C04A serial EEPROM (for configuration storage), the Swiss Army Timer can function as an on delay timer, off delay timer, one-shot, re-triggerable one-shot, astable multivibrator, and enabled astable multivibrator. Its event counter and auto-resetting event counter modes can replicate the function of an electromechanical counter as well. It uses two inputs (a timer input and timer reset input) and has one output. The two inputs and the output can be programmed to be either active high or active low, and the active edge of the timer input can be programmed to be either low-to-high or high-to-low. Time intervals can be programmed from 0.1 seconds to 6553.5 seconds, and terminal counts can be programmed from 1 to 65535.

MODES OF OPERATION

The Swiss Army Timer can operate in one of the following eight modes:

- **Mode 1** - On delay timer: When the timer input goes active, the output waits for N seconds before going active. When the timer input goes inactive, so does the output.
- **Mode 2** - Off delay timer: When the timer input goes active, the output goes active. When the timer input goes inactive, the output waits for N seconds before going inactive.
- **Mode 3** - One shot: When an active transition of the timer input occurs, the output goes active for N seconds. If another active transition of the timer input occurs while the output is active, it is ignored.
- **Mode 4** - Retriggerable one shot: This mode is similar to the one shot mode except if an active transition of the timer input is detected while the output is active, the output will remain active for N seconds after detection of the transition.
- **Mode 5** - Astable multivibrator: This mode replicates the action of an oscillator. The output goes active for N seconds, and then goes inactive for M seconds. This sequence repeats indefinitely.
- **Mode 6** - Enabled astable multivibrator: This mode is similar to the astable multivibrator mode except the sequence is reset and the output is held inactive if the timer input is not active.
- **Mode 7** - Event counter: Each active transition of the timer input is counted. When this count reaches N counts, the output goes active and remains active until the timer reset input becomes active. When the timer reset input goes inactive, the accumulated count goes back to zero. The count is reset any time the timer reset input goes active.
- **Mode 8** - Auto-reset event counter: This mode is similar to the event counter except, the next active transition of the timer input, after the terminal count is reached, will automatically reset the counter and deactivate the output. The timer reset input can still be used to manually reset the accumulated count.

Microchip Technology Incorporated, has been granted a non-exclusive, worldwide license to reproduce, publish and distribute all submitted materials, in either original or edited form. The author has affirmed that this work is an original, unpublished work and that he/she owns all rights to such work. All property rights, such as patents, copyrights and trademarks remain with author.

Electromechanical Timer Replacement

HARDWARE

The Swiss Army Timer circuit consists of; a PIC12C508 8-pin microcontroller, a 24C04A serial EEPROM, an output indicator LED, some resistors, some decoupling capacitors, and several connection points. The timer input is connected to the GP5 I/O pin, the timer reset input is connected to the GP4 I/O pin, and the timer output is tied to the GP2 I/O pin. An HLMP-4700 low current LED is connected to the timer output via a 1 Kohm resistor and serves as an output high indicator. The PIC12C508 is configured to use the internal 4 MHz RC oscillator, and the GP3/MCLR pin is programmed to function as a MCLR input. For non-volatile storage of the operation configuration, a 24C04A serial EEPROM is used. The SDA pin of the EEPROM is tied to the GP1 I/O pin of the PIC12C508, and the SCL pin of the EEPROM is tied to the GP0 pin of the PIC12C508. The CPU reset, EEPROM SDA, and EEPROM SCL lines are brought out as connection points. This is to allow for re-programming the timer with the EEPROM in-circuit. A PC program for in-circuit configuration of the Swiss Army Timer is being worked on, but is not yet complete. For testing, it was necessary to remove the EEPROM from the circuit and use a PROM programmer to change the timer configuration.

Since this circuit was meant to be generic, all I/O was left as logic level. No power supply circuit was included in this circuit for the same reason; thus, an external +5V supply is necessary to power the circuit.

SOFTWARE

The software consists of an initialization block, a main loop, debouncer state machines for each of the inputs, a mode sequencer, a small state machine for each of the operational modes available, several utility subroutines, and subroutines for accessing the EEPROM.

KEY VARIABLES

- **Mode state:** This one byte variable is used by the sequencer mode state machines, and can vary in value from 0 to 2. Some state machines only have two states, whereas most have three states.
- **Target counter:** This two byte variable is used for timing/counting purposes. When in a timer mode, if this value is greater than zero, it is decremented by the time handler in the main loop every 0.1 seconds. Therefore, to setup a timed interval, the mode state machine initializes this variable to the number of tenths of a second to time. When the variable goes to zero, the time interval has elapsed. When in a counter mode, the counter mode state machines directly initialize and decrement this variable.
- **Rollover counter:** This two byte variable is used to determine when a tenth of a second of time has elapsed. It is initialized to 391, which corresponds to the number of RTCC rollovers per one tenth of a second. Whenever the RTCC rolls over, this counter is decremented. When it reaches zero, a tenth of a second of time has elapsed since it was initialized. It is re-initialized, and the process repeats indefinitely.
- **Timer input debounce state:** This one byte variable is used by the timer input debounce state machine, and can vary in value from 0 to 3. Whenever this variable is 0 or 1, the input is considered to be low, and if it is 2 or 3, the input is considered to be high.
- **Timer reset input debounce state:** This one byte value is used by the timer reset input debounce state machine. It operates similarly to the timer input debounce state variable.

Electromechanical Timer Replacement

CONFIGURATION BLOCK

These values, which are read from the EEPROM and are stored in contiguous memory locations, are used to configure the operation of the Swiss Army Timer. They are as follows:

- **Mode:** This one byte value determines which of the eight available modes the Swiss Army Timer operates in. These values are defined as follows:
 - 0 = on delay timer
 - 1 = off delay timer
 - 2 = one-shot
 - 3 = retriggerable one-shot
 - 4 = astable multivibrator
 - 5 = enabled astable multivibrator
 - 6 = event counter
 - 7 = auto-resetting event counter
- **Time parameter 1:** This two byte value is used as the delay/pulse width/terminal count value. In the astable modes, it is used as the on time value. It can vary from 1 to 65535.
- **Time parameter 2:** This two byte value is used by the astable modes as the off time value. It can vary from 1 to 65535.
- **Timer options:** This one byte value is used to select the active levels of the timer input, reset input, timer output, and timer input active edge. Four of the bits in this value are used for this purpose, and they are as follows:
 - Bit 0: If high, the timer input is configured as active low. Otherwise, the timer input is active high.
 - Bit 1: If high, the timer output is configured as active low. Otherwise, the timer output is active high.
 - Bit 2: If high, the timer reset input is configured as active low. Otherwise, the timer reset input is active high.
 - Bit 3: If high, the timer input active edge is configured as high to low. Otherwise, the timer input active edge is low to high.

INITIALIZATION BLOCK

Invoked on a CPU reset, this section trims the on-board oscillator, sets up the OPTION register, initializes the GPIO register as well as the TRIS register, initializes the file registers (RAM) used by the program, and reads the configuration from the EEPROM.

MAIN LOOP

After initialization, the main loop runs indefinitely. Each pass through the main loop, we:

- Check to see if the RTCC has rolled over (this will occur every 256 microseconds). If it has, we decrement the rollover counter and the debounce counter. When the rollover counter is zero (this occurs every 0.1 seconds), we re-initialize it to 391 ($0.1 \text{ seconds} / 256 \text{ microseconds} = 390.625$), and if the timer target counter is not equal to zero and we are not in an event counter mode, we decrement it as well.
- If the debounce counter is zero, we re-initialize it to 20 ($20 * 256 \text{ microseconds} = 5.12 \text{ milliseconds}$) and call the timer input and reset input state machine routines. Based on the current state of each of these state machines and the timer option flags in the configuration, the timer input and reset input flags in the flags register are set to either active or inactive. If an active transition of the timer input has occurred, the timer input edge flag in the flags register is also set.
- The mode sequencer routine is called. Based on the configured mode of operation, the sequencer jumps to the appropriate mode state machine routine. Each of these state machines determine whether the timer output should be active or inactive based on the current mode sequencer state. This is done by either setting or clearing the output active flag in the flags register. They also determine whether to switch states based on the timer input and reset input, and the value of the timer target counter.
- The timer output is set either high or low based on the output active flag in the flags register and the active output level setting in the timer option flags in the configuration.
- The watchdog timer is reset, and we jump back to the beginning of the loop.

DEBOUNCER STATE MACHINES

There are two of these 4-state machines. One is for the timer input, and one is for the timer reset input. For a given input level to be considered valid, the state of the corresponding input pin must remain the same for at least two passes through the state machine routine. This helps minimize false input triggering if mechanical switches are used.

MODE SEQUENCER

The mode sequencer routine is nothing more than a jump table. Based on the configured mode of operation, the sequencer causes program execution to jump to the appropriate mode state machine. Each one of these state machines jump back to the end of the mode sequencer when they are finished.

Electromechanical Timer Replacement

MODE STATE MACHINES

There are eight small mode state machines, one for each of the available modes of operation. Each one of these state machines determine the state of the timer output based on its current state.

They also determine whether or not to change state based on the timer inputs, the timer target counter value. When a state change occurs, the value of the timer target counter may also be updated.

FIGURE 1: ON DELAY MODE STATE DIAGRAM

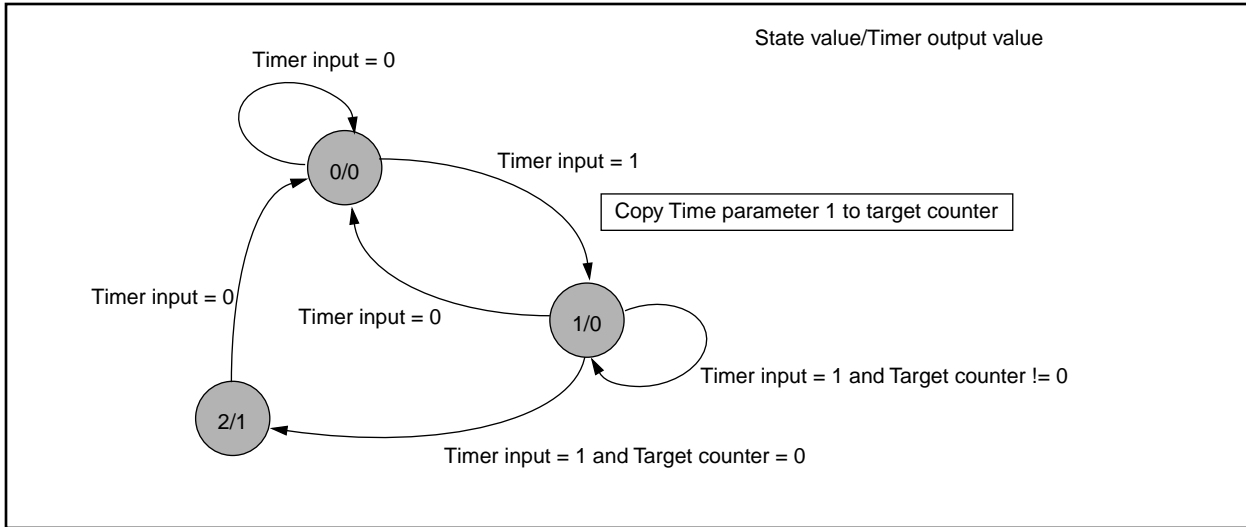
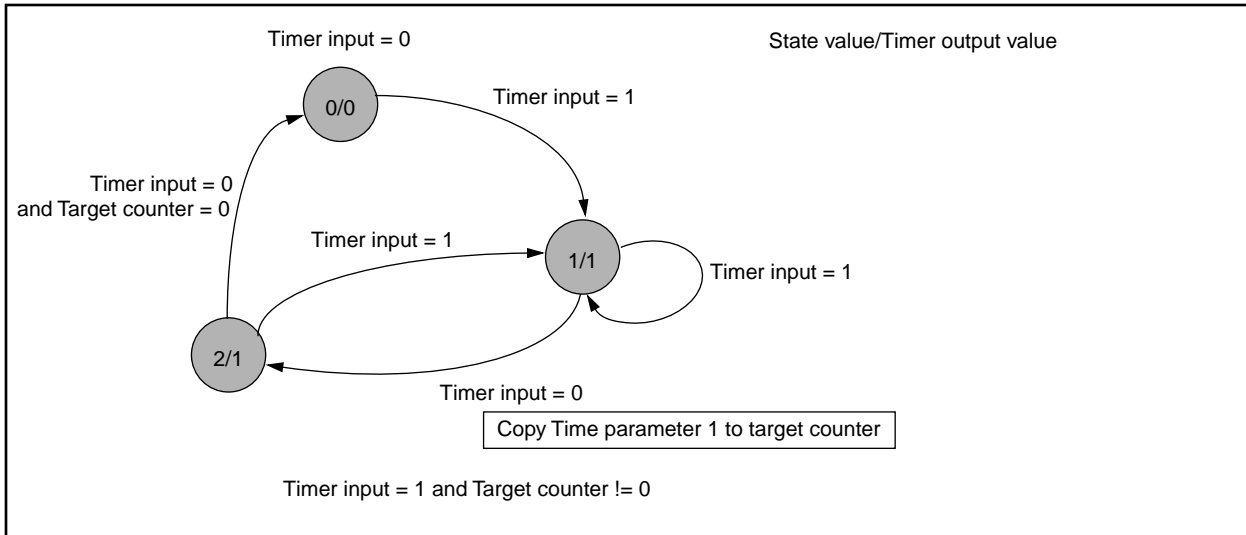


FIGURE 2: OFF DELAY MODE STATE DIAGRAM



Electromechanical Timer Replacement

FIGURE 3: ONE SHOT MODE STATE DIAGRAM

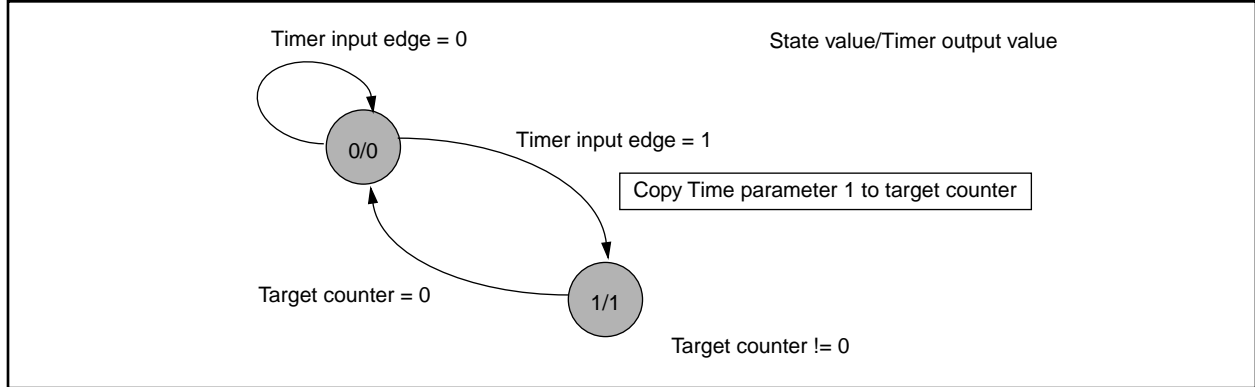


FIGURE 4: RETRIGGERABLE ONE SHOT MODE STATE DIAGRAM

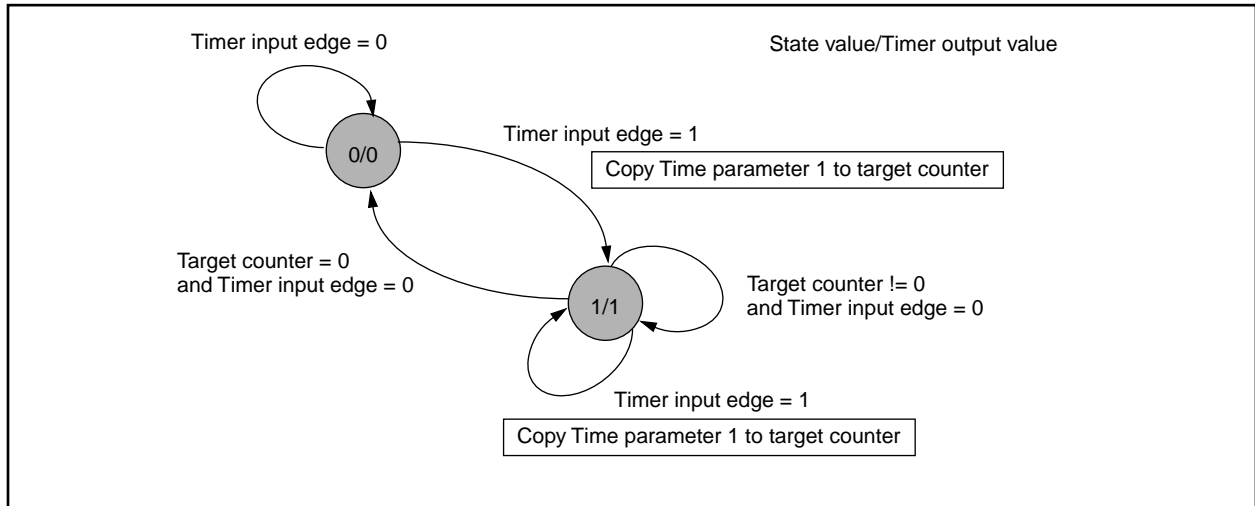
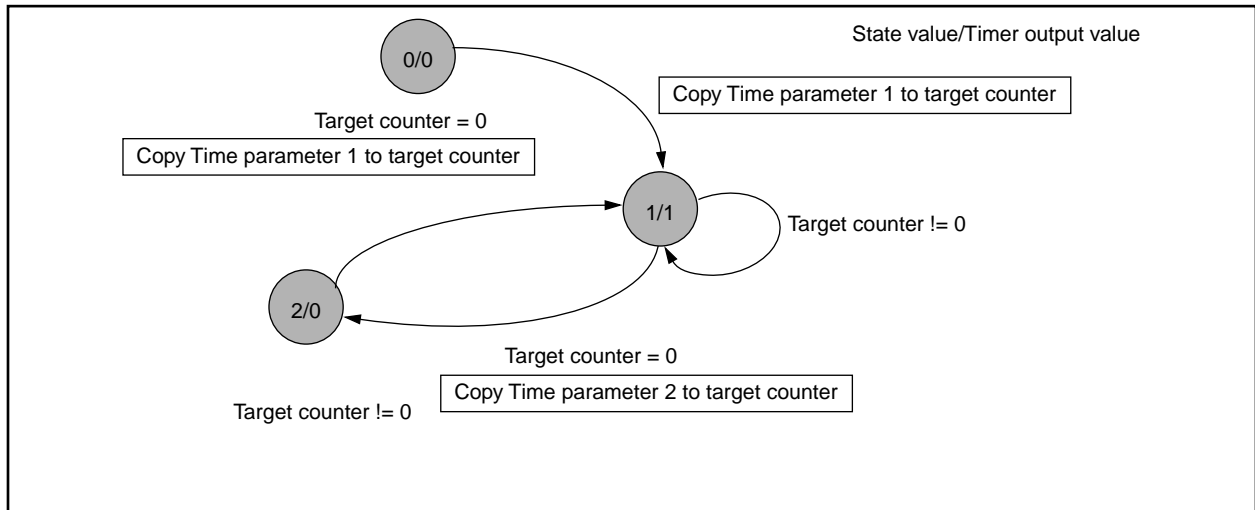


FIGURE 5: ASTABLE MULTIVIBRATOR MODE STATE DIAGRAM



Electromechanical Timer Replacement

FIGURE 6: ENABLED ASTABLE MULTIVIBRATOR MODE STATE DIAGRAM

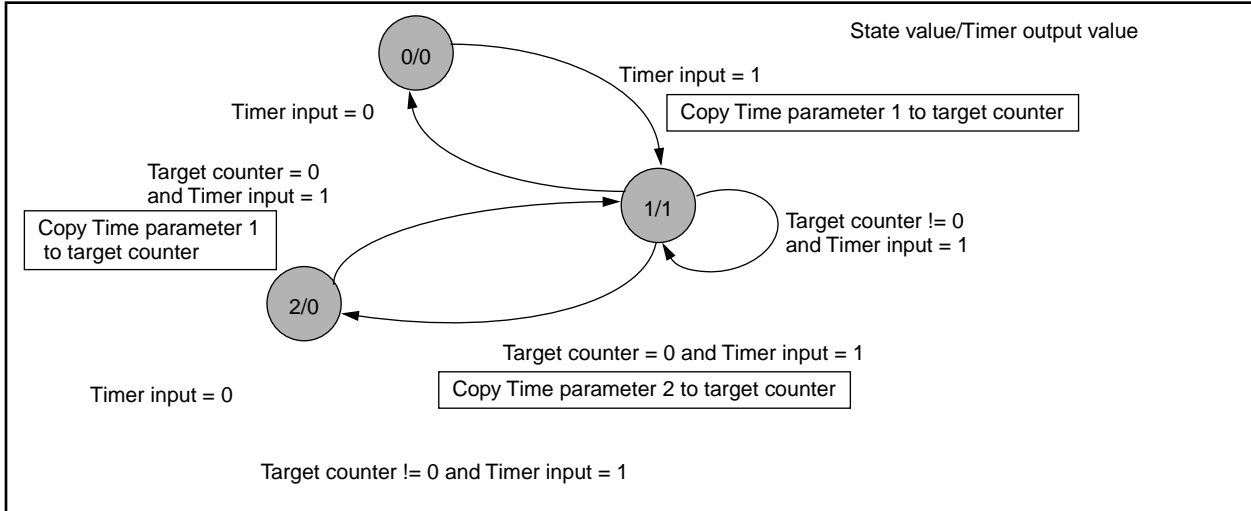


FIGURE 7: EVENT COUNTER MODE STATE DIAGRAM

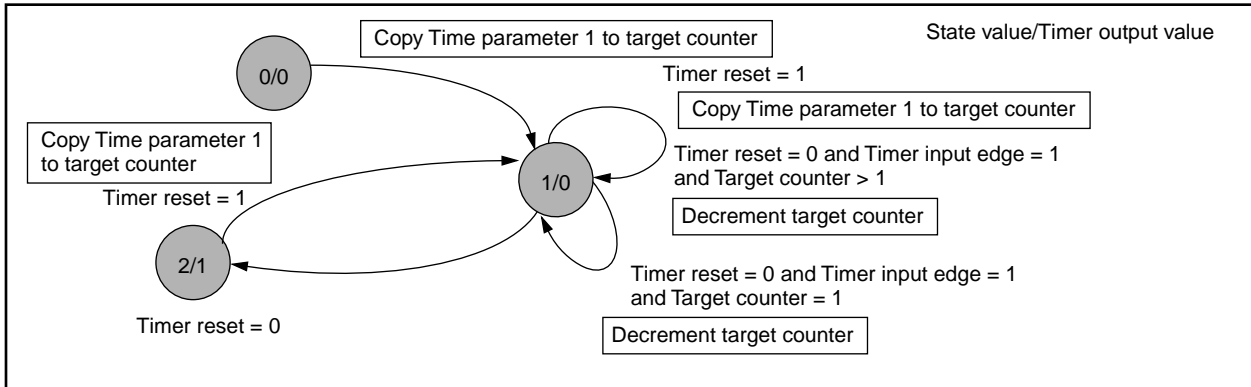
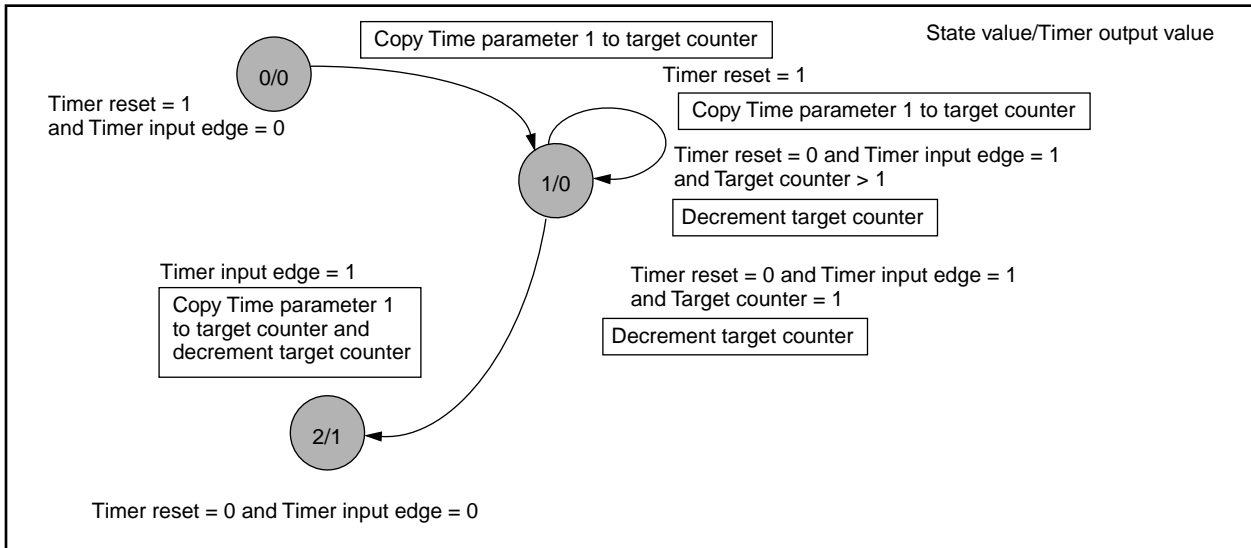
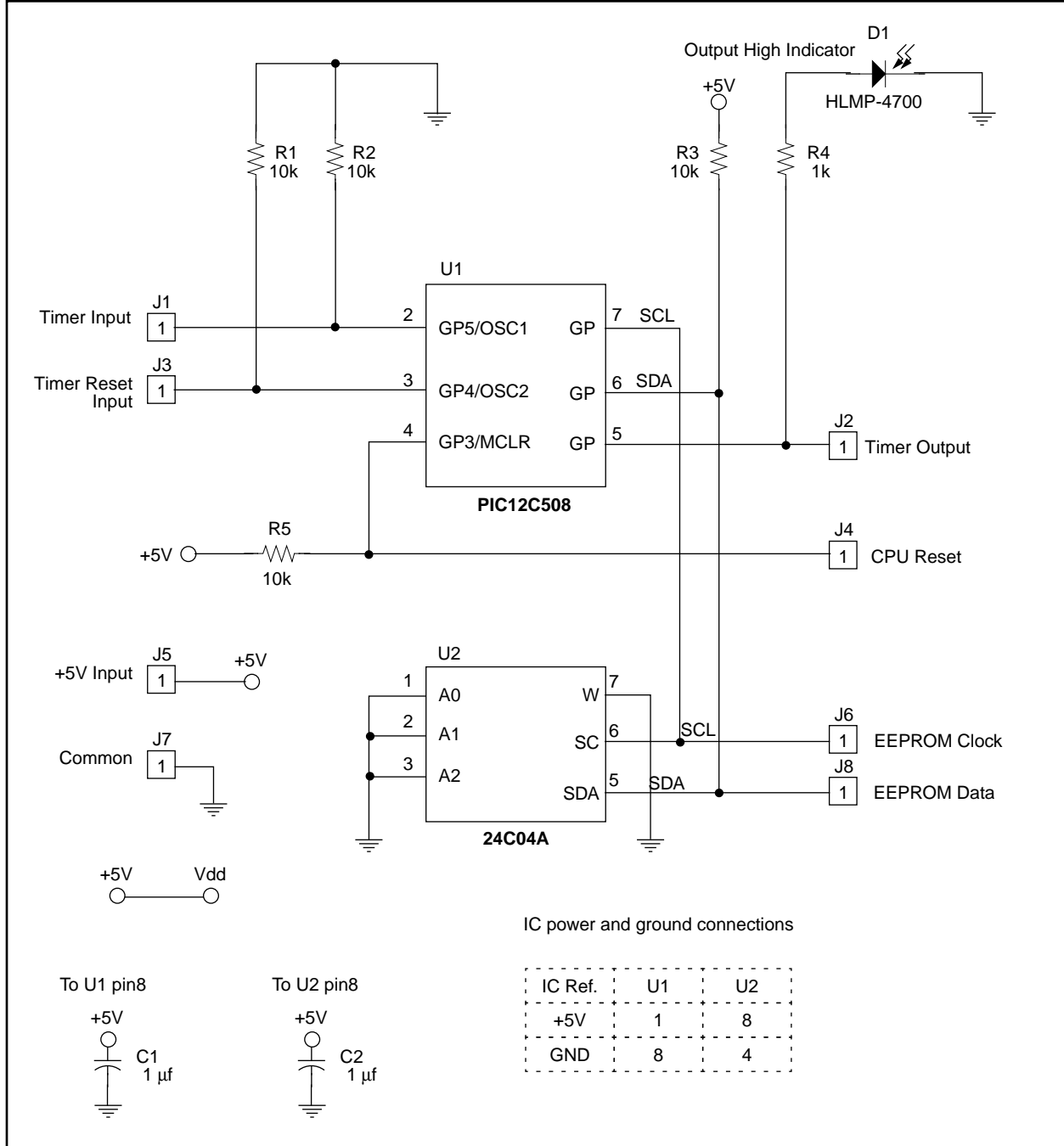


FIGURE 8: AUTO-RESET EVENT COUNTER MODE STATE DIAGRAM



Electromechanical Timer Replacement

FIGURE 9: SCHEMATIC



MICROCHIP TOOLS USED

Hardware Development Tools:

The PICMASTER emulator with a PIC16C54 POD

Assembler/Compiler Version:

MPLAB 3.22.00 with MPASM 1.50

Electromechanical Timer Replacement

APPENDIX A: SOURCE CODE

The PICMASTER emulator with a 16C54 pod was used to debug the PIC16C54 test version.

Assembler/Compiler version:

MPLAB 3.22.00 development software with MPASM version 1.50

Almost all debugging was done running in simulator mode.

```
*****
;*   Swiss Army Timer Project           *
;*   using the PIC12C508 Version 1.0    *
;*                                     *
;*   Version 1.0 written 5/20/1997     *
;*   by Michael Kirkhart               *
;*   Release date: 5/30/1997           *
;*                                     *
*****
list      p=12C508      ;specifies 12C508 microcontroller
list      r=DEC         ;specifies decimal radix as default
list      x=ON         ;specifies to expand macros in listing
errorlevel 1           ;print warnings and errors only in list file

*****
;* General system info *
*****
;
;Instruction clock frequency = 4MHz
;Non-branching instruction execution time = 1 microsecond
;Configuration word settings: Watchdog timer = ON
;                               Code Protect = OFF
;                               MCLR pin = ENABLED
;                               Oscillator = INTERNAL RC
__config      0xff6

*****
;* CPU Register equates *
*****
IND0    equ    00      ;indirect file register
RTCC    equ    01      ;real time clock/counter
PC      equ    02      ;program counter
STATUS  equ    03      ;status register
FSR     equ    04      ;file select register (pointer)
OSCCAL  equ    05      ;on chip oscillator calibration register
GPIO    equ    06      ;general purpose I/O register

*****
;* Status register bit definitions *
*****
CARRY    equ    0      ;carry/!borrow flag
DCARRY   equ    1      ;BCD carry/!borrow flag
ZERO     equ    2      ;zero flag
PDOWN    equ    3      ;powerdown flag
TIMEOUT  equ    4      ;watchdog timeout flag

*****
;* GPIO bit definitions *
*****
SCL      equ    0      ;EEPROM serial clock (O)
SDA      equ    1      ;EEPROM serial data (I/O)
TMRROUT  equ    2      ;Timer output (O)
TMRRST   equ    4      ;Timer reset (I)
TMRIN    equ    5      ;Timer input (I)

*****
;* Equates for register files (variables)*
*****
```

Electromechanical Timer Replacement

```
*****
; configuration block from EEPROM (must be in contiguous memory locations)
mode          equ      0x07      ;timer mode from EEPROM
param1L      equ      0x08      ;parameter 1 low byte from EEPROM
param1H      equ      0x09      ;parameter 1 high byte from EEPROM
param2L      equ      0x0a      ;parameter 2 low byte from EEPROM
param2H      equ      0x0b      ;parameter 2 high byte from EEPROM
tmropt       equ      0x0c      ;option flags from EEPROM

; file registers used in timer handler
oldRTCC      equ      0x0d      ;keeps track of RTCC value from last run through loop
rollL        equ      0x0e      ;RTCC rollover counter low byte
rollH        equ      0x0f      ;RTCC rollover counter high byte

; miscellaneous state variables, target counters, etc.
inDbnc       equ      0x10      ;Timer input debouncer state
rstDbnc       equ      0x11      ;Reset input debouncer state
dbncTmr      equ      0x12      ;Debounce check timer
tgtL         equ      0x13      ;target counter low byte
tgtH         equ      0x14      ;target counter high byte
flags        equ      0x15      ;timer status flags
modest       equ      0x16      ;state of selected timer mode
temp         equ      0x17      ;temporary storage register

; file registers used by EEPROM read routines
eeprom       equ      0x18      ;bit buffer
datai        equ      0x19      ;data input register
txbuf        equ      0x1a      ;transmit buffer
count        equ      0x1b      ;bit counter
bcount       equ      0x1c      ;byte counter

*****
;* Miscellaneous equates (constants) *
*****

;GPIO initialization values
GPINIT       equ      00000010b   ;GPIO initial value (SDA line high, all others low)
GPTRIS       equ      00111010b   ;GPIO TRIS register initial value
SDAINP       equ      00111010b   ;GPIO TRIS register value when SDA line to EEPROM
; needs to be an input
SDAOUT       equ      00111000b   ;GPIO TRIS register value when SDA line to EEPROM
; needs to be an output

;mode constants
ONDLY        equ      0           ;mode 0 = on delay timer
OFFDLY       equ      1           ;mode 1 = off delay timer
ONESHT       equ      2           ;mode 2 = non-retriggerable one-shot
RONESHT      equ      3           ;mode 3 = retriggerable one-shot
ASTABL       equ      4           ;mode 4 = astable multivibrator
EASTABL      equ      5           ;mode 5 = astable multivibrator with enable
COUNTR       equ      6           ;mode 6 = event counter
COUNTRA      equ      7           ;mode 7 = event counter with auto-reset

;option flags bit values
LOWIN        equ      0           ;active low input flag
LOWOUT       equ      1           ;active low output flag
LOWRST       equ      2           ;active low reset flag
TRLEDG       equ      3           ;trailing input edge active flag

;flags register bit values
INPHI        equ      0           ;timer input high
OUTH1        equ      1           ;timer output high
RSTHI        equ      2           ;reset input high
EDGON        equ      3           ;timer input transistion detected
INPON        equ      4           ;timer input active
```

Electromechanical Timer Replacement

```

    OUTON      equ    5      ;timer output active
    RSTON      equ    6      ;reset input active

;bit defines for EEPROM routines
    do         equ    6      ;eeprom output bit
    di         equ    7      ;eeprom input bit

;timer constants
    DBNCTM     equ    20     ;debounce timer interval (20 * 256 microseconds =
5.12milliseconds)
    TENTHL     equ    0x87   ;tenth second rollover count (low byte)
    TENTHH     equ    0x01   ;tenth second rollover count (high byte)

;*****
;* Macro definitions      *
;*****

CLC      macro                ;this macro will clear the C flag
    bcf   STATUS,CARRY
endm

SEC      macro                ;this macro will set the C flag
    bsf   STATUS,CARRY
endm

SCC      macro                ;used after an instruction that affects the C
    btfsc STATUS,CARRY        ; flag, this macro will skip the next
endm                                           ; instruction if the C flag is clear

SCS      macro                ;used after an instruction that affects the C
    btfss STATUS,CARRY        ; flag, this macro will skip the next
endm                                           ; instruction if the C flag is set

SLT      macro                ;used after a subtract instruction, this macro
    btfsc STATUS,CARRY        ; will skip the next instruction if the result
endm                                           ; of the subtraction is < 0

SGE      macro                ;used after a subtract instruction, this macro
    btfss STATUS,CARRY        ; will skip the next instruction if the result
endm                                           ; of the subtraction is >= 0

SEQ      macro                ;used after an instruction that affects the Z
    btfss STATUS,ZERO         ; flag, this macro will skip the next
endm                                           ; instruction if a result is zero

SNE      macro                ;used after an instruction that affects the Z
    btfsc STATUS,ZERO         ; flag, this macro will skip the next
endm                                           ; instruction if a result is non-zero

;*****
;* Start of program      *
;*****
; actual reset vector - instruction at address 0x1fff was movlw XX, where
; XX is the calibration value to be copied into the OSCCAL register

    org      0                ;start of program memory
    movwf   OSCCAL            ;calibrate on-chip oscillator
    goto    start             ;jump to start of program

;*****
;* Subroutines          *
;* These must be located in the *
;* lower 256 bytes of program *
;* memory                *
;*****
```

Electromechanical Timer Replacement

```
*****
;* 24C04 EEPROM read routines *
;* Modified versions of 24CXX *
;* routines from Microchip *
;* AN567 *
*****

*****
; Start Bit Subroutine *
; this routine generates a start bit *
; (Low going data line while clock is high) *
*****
bStart
    bsf        GPIO,SDA        ;make sure data is high
    movlw     SDAOUT
    tris      GPIO            ;set data and clock lines for output
    bcf      GPIO,SCL        ;make sure clock is low
    nop
    bsf      GPIO,SCL        ;set clock high
    call     eeDelay         ;wait for a few cycles
    bcf      GPIO,SDA        ;data line goes low during
                                ; high clock for start bit
    call     eeDelay         ;wait for a few cycles
    bcf      GPIO,SCL        ;start clock train
    call     eeDelay         ;wait for a few cycles
    retlw    0              ;return from subroutine

*****
; Stop Bit Subroutine *
; This routine generates a stop bit *
; (High going data line while clock is high) *
*****
bStop
    movlw     SDAOUT        ;
    tris      GPIO            ;set data/clock lines as outputs
    bcf      GPIO,SDA        ;make sure data line is high
    call     eeDelay         ;wait for a few cycles
    bsf      GPIO,SCL        ;set clock high
    call     eeDelay         ;wait for a few cycles
    bsf      GPIO,SDA        ;data goes high while clock high
                                ;for stop bit
    call     eeDelay         ;wait for a few cycles
    bcf      GPIO,SCL        ;set clock low again
    call     eeDelay         ;wait for a few cycles
    retlw    0              ;return from subroutine

*****
; BITOUT routine takes one bit of data in 'do' and *
; transmits it to the serial EE device *
*****
bitOut
    movlw     SDAOUT        ;set data, clock as outputs
    tris      GPIO            ;
    btfss    eeprom,do      ;check for stat of data bit to xmit
    goto     bitlow         ;
    bsf      GPIO,SDA        ;set data line high
    goto     clkout         ;go toggle the clock
bitlow    bcf      GPIO,SDA        ;output a low bit
clkout    bsf      GPIO,SCL        ;set clock line high
    nop
    nop
    bcf      GPIO,SCL        ;return clock line low
    retlw    0              ;return from subroutine
```

Electromechanical Timer Replacement

```
;*****
; eeDelay routine generates a small delay that is      *
; used by the various EEPROM routines                 *
;*****
eeDelay
    nop                ;
    nop                ;
    retlw              0                ;

;*****
; BITIN routine reads one bit of data from the        *
; serial EE device and stores it in 'di'             *
;*****
bitIn    bsf          eeeprom,di        ;assume input bit is high
        movlw        SDAINP            ;make sdata an input line
        tris         GPIO              ;
        bsf          GPIO,SCL          ;set clock line high
        nop          ;wait a few cycles
        nop
        btfss       GPIO,SDA          ;read the data bit
        bcf          eeeprom,di        ;input bit was low, set 'di' accordingly
        bcf          GPIO,SCL          ;set clock line low
        retlw        0                ;return from subroutine
;
;*****
; Transmit Data Subroutine                            *
; This routine takes the byte of data stored in the  *
; 'data0' register and transmits it to the serial EE device. *
; It will then send 1 more clock to the serial EE for the *
; acknowledge bit. If the ack bit from the part was low *
; then the transmission was successful. If it is high, then *
; the device did not send a proper ack bit and the ack *
; fail LED will be turned on.                        *
;*****
tx
        movlw        8                ;
        movwf        count            ;

txLoop   bcf          eeeprom,do        ;assume bit out is low
        btfsc        txbuf,7          ;is bit out really low?
        bsf          eeeprom,do        ;no, set it high
        call         bitOut           ;send the bit to serial EE
        rlf          txbuf            ;rotate txbuf left
        decfsz       count            ;8 bits done?
        goto         txLoop           ;no - go again
        call         bitIn            ;read ack bit
        retlw        0                ;return from subroutine
;
;*****
; Receive data Routine                               *
; This routine reads one byte of data from the part *
; into the 'data1' register. It then sends a high *
; ack bit to indicate that no more data is to be read *
;*****
rx
        movlw        8                ;set # bits to 8
        movwf        count            ;
        clrf         data1            ;clear input register
        bcf          STATUS,CARRY      ;make sure carry bit is low
rxLoop   rlf          data1            ;rotate data1 1 bit left
        call         bitIn            ;read a bit
        btfsc       eeeprom,di        ;
        bsf          data1,0          ;set bit 0 if necessary
        decfsz       count            ;8 bits done?
        goto         rxLoop           ;no, do another
        retlw        0                ;return from subroutine
```

Electromechanical Timer Replacement

```
*****
;* MoveP1ToTgt *
*****
; This routine moves the timer parameter 1 values read from
; the configuration EEPROM into the target timer file registers.
; This routine is used by the various state handlers in the mode sequencer.

MoveP1ToTgt
    movf      param1L,w      ;initialize
    movwf    tgtL           ; target
    movf      param1H,w      ; counter
    movwf    tgtH           ; value
    goto     InitRollOver   ;re-initialize rollover counter
                                ; (return instruction executed at end of InitRollOver)

*****
;* MoveP2ToTgt *
*****
; This routine moves the timer parameter 2 values read from
; the configuration EEPROM into the target timer file registers.
; This routine is used by the various state handlers in the mode sequencer.

MoveP2ToTgt
    movf      param2L,w      ;initialize
    movwf    tgtL           ; target
    movf      param2H,w      ; counter
    movwf    tgtH           ; value
    goto     InitRollOver   ;re-initialize rollover counter
                                ; (return instruction executed at end of InitRollOver)

*****
;* InitRollOver *
*****
; This routine initializes the rollover counter with the value that
; represents .1 second

InitRollOver
    movlw    TENTHL         ;load
    movwf    rollL          ; rollover
    movlw    TENTHH         ; counter
    movwf    rollH          ;
    retlw    0              ;return from subroutine

*****
;* IsTgtZero *
*****
; This routine checks to see if the target timer is equal
; to zero. If it is, it returns with the zero flag in status register set. Otherwise,
; the zero flag is cleared. This routine is used by the various state handlers in the
; mode sequencer.

IsTgtZero
    movlw    0              ;is upper byte of target timer
    iorwf    tgtH,w         ; = 0?
    SEQ                                           ;if yes, skip
    goto     TgtNotZero     ;if not, target timer not zero - branch
    iorwf    tgtL,w         ;is lower byte of target timer = 0? If so, zero flag in
                                ; status register will be set

TgtNotZero
    retlw    0              ;return from subroutine (result in zero flag in status reg)

*****
;* DecrTgt *
*****
; This routine decrements the target counter. It is used in the timer handler and the
; counter mode handlers in the mode sequencer.
```

Electromechanical Timer Replacement

```
DecrTgt
    movlw    1           ;subtract 1 from
    subwf   tgtL        ; tgtL (lower byte of target)
    SGE                    ;did borrow occur?
    subwf   tgtH        ;if so, subtract 1 from tgtH (upper byte of target)
    retlw   0           ;return from subroutine

;*****
;* doTmrInState
;* This subroutine runs the debounce state machine
;* for the Timer Input. It is called periodically by
;* the main program loop.
;*****
doTmrInState
    movf    inDbnc,w    ;get current timer input debounce state
    addwf   PC          ;add it to the program counter to jump to
                        ; appropriate state handler

    goto    TInSt0     ;inDbnc = 0 handler
    goto    TInSt1     ;inDbnc = 1 handler
    goto    TInSt2     ;inDbnc = 2 handler
    goto    TInSt3     ;inDbnc = 3 handler

;TimerIn state = 0 handler
TInSt0  btfss   GPIO,TMRIN ;is Timer input high?
        goto    TInEnd    ;if no - stay in state 0
        incf    inDbnc    ;if yes - move to state 1
        goto    TInEnd    ;go to TInEnd

;TimerIn state = 1 handler
TInSt1  btfss   GPIO,TMRIN ;is Timer input high?
        goto    TInSt1a   ;if no - go to TInSt1a
        incf    inDbnc    ;if yes - move to state 2
        goto    TInEnd    ;go to TInEnd
TInSt1a clrf    inDbnc    ;move back to state 0
        goto    TInEnd    ;go to TInEnd

;TimerIn state = 2 handler
TInSt2  btfsc   GPIO,TMRIN ;is Timer input low?
        goto    TInEnd    ;if no - go to TInEnd (stay in state 2)
        incf    inDbnc    ;if yes - move to state 3
        goto    TInEnd    ;go to TInEnd

;TimerIn state = 3 handler
TInSt3  btfsc   GPIO,TMRIN ;is Timer input low?
        goto    TInSt3a   ;if no - go to TInSt3a
        clrf    inDbnc    ;if yes - move to state 0
        goto    TInEnd    ;go to TInEnd
TInSt3a decf    inDbnc    ;move back to state 2
        goto    TInEnd    ;go to TInEnd

TInEnd  retlw   0           ;return from subroutine

;*****
;* doRstInState
;* This subroutine runs the debounce state machine
;* for the Reset Input. It is called periodically by
;* the main program loop.
;*****
doRstInState
    movf    rstDbnc,w    ;get current reset input debounce state
    addwf   PC          ;add it to the program counter to jump to the
                        ; appropriate state handler

    goto    RstSt0     ;rstDbnc = 0 state handler
    goto    RstSt1     ;rstDbnc = 1 state handler
    goto    RstSt2     ;rstDbnc = 2 state handler
    goto    RstSt3     ;rstDbnc = 3 state handler
```

Electromechanical Timer Replacement

```
;ResetIn state = 0 handler
RstSt0 btfss    GPIO,TMRST    ;is timer reset input high?
      goto     RstEnd        ;if no - stay in state 0
      incf     rstDbnc        ;if yes - move to state 1
      goto     RstEnd        ;go to RstEnd

RstSt1 btfss    GPIO,TMRST    ;is timer reset input high?
      goto     RstSt1a       ;if no - go to RstSt1a
      incf     rstDbnc        ;if yes - move to state 2
      goto     RstEnd        ;go to RstEnd

RstSt1a clrf     rstDbnc       ;move back to state 0
      goto     RstEnd        ;go to RstEnd

RstSt2 btfsc    GPIO,TMRST    ;is timer reset input low?
      goto     RstEnd        ;if no - go to RstEnd (stay in state 2)
      incf     rstDbnc        ;if yes - move to state 3
      goto     RstEnd        ;go to RstEnd

RstSt3 btfsc    GPIO,TMRST    ;is timer reset input low?
      goto     RstSt3a       ;if no - go to RstSt3a
      clrf     rstDbnc        ;if yes - move to state 0
      goto     RstEnd        ;go to RstEnd

RstSt3a decf     rstDbnc       ;move back to state 2
      goto     RstEnd        ;go to RstEnd

RstEnd retlw    0             ;return from subroutine

;;doSeqState
;*****
;* doSeqState *
;* This subroutine runs the main sequencer state *
;* machine. It is called periodically by *
;* the main program loop. *
;*****
doSeqState
    movf     mode,w           ;get configuration mode value
    addwf    PC               ;add it to the program counter to jump to the
                              ; appropriate mode handler
    goto     doOnDelay        ;on delay timer handler
    goto     doOffDelay       ;off delay timer handler
    goto     doOneShot        ;one shot handler
    goto     doROneShot       ;retriggerable one shot handler
    goto     doAstable        ;astable multivibrator handler
    goto     doEAstable       ;enabled astable multivibrator handler
    goto     doCounter        ;event counter handler
    goto     doACounter       ;auto resetting event counter handler

;*****
;* On Delay mode handler *
;*****
doOnDelay
    movf     modest,w        ;get current sequencer state value
    addwf    PC               ;add it to the program counter to jump to the
                              ; appropriate mode handler
    goto     OnDly0          ;On delay state 0 handler
    goto     OnDly1          ;On delay state 1 handler
    goto     OnDly2          ;On delay state 2 handler

;*****
;* Off Delay mode handler *
;*****
doOffDelay
    movf     modest,w        ;get current sequencer state value
```

Electromechanical Timer Replacement

```
        addwf    C                ;add it to the program counter to jump to the
                                   ; appropriate mode handler
        goto    OffDly0           ;Off delay state 0 handler
        goto    OffDly1           ;Off delay state 1 handler
        goto    OffDly2           ;Off delay state 2 handler

;*****
;* Non-retriggerable one-shot mode handler      *
;*****
doOneShot
        movf    modest,w         ;get current sequencer state value
        addwf   PC                ;add it to the program counter to jump to the
                                   ; appropriate mode handler
        goto    OneShot0          ;one shot state 0 handler
        goto    OneShot1          ;one shot state 1 handler

;*****
;* Retriggerable one-shot mode handler        *
;*****
doROneShot
        movf    modest,w         ;get current sequencer state value
        addwf   PC                ;add it to the program counter to jump to the
                                   ; appropriate mode handler
        goto    OneShot0          ;retriggerable one shot state 0 handler
                                   ; (same as non-retriggerable
                                   ; one shot state 0 handler)
        goto    ROneShot1         ;retriggerable one shot state 1 handler

;*****
;* Astable mode handler                      *
;*****
doAstable
        movf    modest,w         ;get current sequencer state value
        addwf   PC                ;add it to the program counter to jump to the
                                   ; appropriate mode handler
        goto    Astable0          ;astable state 0 handler
        goto    Astable1          ;astable state 1 handler
        goto    Astable2          ;astable state 2 handler

;*****
;* Enabled astable mode handler              *
;*****
doEAstable
        movf    modest,w         ;get current sequencer state value
        addwf   PC                ;add it to the program counter to jump to the
                                   ; appropriate mode handler
        goto    EAstable0         ;astable state 0 handler
        goto    EAstable1         ;astable state 1 handler
        goto    EAstable2         ;astable state 2 handler

;*****
;* Event counter mode handler                *
;*****
doCounter
        movf    modest,w         ;get current sequencer state value
        addwf   PC                ;add it to the program counter to jump to the
                                   ; appropriate mode handler
        goto    Counter0          ;event counter state 0 handler
        goto    Counter1          ;event counter state 1 handler
        goto    Counter2          ;event counter state 2 handler

;*****
;* Auto-reset event counter mode handler     *
;*****
doACounter
        movf    modest,w         ;get current sequencer state value
```

Electromechanical Timer Replacement

```
    addwf    PC                ;add it to the program counter to jump to the
                                ; appropriate mode handler
    goto    Counter0          ;auto-reset event counter state 0 handler
                                ; (same as event counter state 0 handler)
    goto    Counter1          ;auto-reset event counter state 1 handler
                                ;(same as event counter state 1 handler)
    goto    ACounter2        ;auto-reset event counter state 2 handler

;*****
;* Main sequencer subroutine return point *
;*****
seqEnd    retlw    0          ;return from subroutine

; on delay state 0 - waiting for input to go active
OnDly0
    bcf     flags,OUTON      ;clear timer output on flag
    btfsc  flags,INPON      ;is timer input active?
    goto   OnDly0a          ;if yes, go to OnDly0a
    goto   seqEnd           ;go to seqEnd
OnDly0a   incf    modest     ;move to mode state 1
    call   MoveP1ToTgt      ;initialize target counter value
    goto   seqEnd           ;go to seqEnd

; on delay state 1 - input active, waiting for target timer to time out
OnDly1
    bcf     flags,OUTON      ;clear timer output on flag
    btfsc  flags,INPON      ;is timer input still active?
    goto   OnDly1a          ;if yes, go to OnDly1a
    clrf   modest          ;otherwise, go back to state 0
    goto   seqEnd           ;go to seqEnd

OnDly1a   call   IsTgtZero    ;is target timer = 0?
    SEQ                      ;
    goto   seqEnd           ;if not, go to seqEnd
    incf   modest          ;otherwise, move to mode state 2
    goto   seqEnd           ;go to seqEnd

OnDly2
    bsf     flags,OUTON      ;set timer output on flag
    btfsc  flags,INPON      ;is timer input still active?
    goto   seqEnd           ;if yes, go to OnDly2a
    clrf   modest          ;otherwise, go back to state 0
    goto   seqEnd           ;go to seqEnd

OffDly0
    bcf     flags,OUTON      ;clear timer output flag
    btfsc  flags,INPON      ;is timer input active?
    goto   OffDly0a         ;if yes, go to OffDly0a
    goto   seqEnd           ;otherwise, goto seqEnd

OffDly0a
    incf   modest          ;move to mode state 1
    goto   seqEnd           ;go to seqEnd

OffDly1
    bsf     flags,OUTON      ;set timer output on flag
    btfsc  flags,INPON      ;is timer input still active?
    goto   seqEnd           ;if yes, go to seqEnd
    incf   modest          ;move to mode state 2
    call   MoveP1ToTgt      ;initialize target counter value
    goto   seqEnd           ;go to seqEnd

OffDly2
    bsf     flags,OUTON      ;set timer output on flag
    btfsc  flags,INPON      ;is timer input active again?
    goto   OffDly2a         ;if so, go to OffDly2a
```

Electromechanical Timer Replacement

```
    call    IsTgtZero      ;is target timer = 0?
    SEQ
    goto    seqEnd        ;if not, go to seqEnd
    clrf   modest        ;otherwise, move to mode state 0
    goto    seqEnd        ;go to seqEnd

OffDly2a
    decf   modest        ;move back to mode state 1
    goto    seqEnd        ;go to seqEnd

OneShot0
    bcf    flags,OUTON    ;clear timer output on flag
    btfsc  flags,EDGON    ;has an active edge been detected?
    goto   OneShot0a     ;if yes, go to OnDly0a
    goto   seqEnd        ;go to seqEnd

OneShot0a
    incf   modest        ;move to mode state 1
    call   MoveP1ToTgt    ;initialize target counter value
    goto   seqEnd        ;go to seqEnd

OneShot1
    bsf    flags,OUTON    ;set timer output on flag
    call   IsTgtZero      ;is target timer = 0?
    SEQ
    goto   seqEnd        ;if not, go to seqEnd
    clrf   modest        ;otherwise, move to mode state 0
    goto   seqEnd        ;go to seqEnd

ROneShot1
    bsf    flags,OUTON    ;set timer output on flag
    btfsc  flags,EDGON    ;have we been retriggered?
    goto   ROneShot1a    ;yes we have - go to ROneShot1a
    call   IsTgtZero      ;is target timer = 0?
    SEQ
    goto   seqEnd        ;if not, go to seqEnd
    clrf   modest        ;otherwise, move to mode state 0
    goto   seqEnd        ;go to seqEnd

ROneShot1a
    call   MoveP1ToTgt    ;initialize target counter value
    goto   seqEnd        ;go to seqEnd

Astable0
    bcf    flags,OUTON    ;clear timer output on flag
    incf   modest        ;move to mode state 1
    call   MoveP1ToTgt    ;initialize target counter value
    goto   seqEnd        ;go to seqEnd

Astable1
    bsf    flags,OUTON    ;set timer output on flag
    call   IsTgtZero      ;is target timer = 0?
    SEQ
    goto   seqEnd        ;if not, go to seqEnd
    incf   modest        ;otherwise, move to mode state 2
    call   MoveP2ToTgt    ;initialize target counter value
    goto   seqEnd        ;go to seqEnd

Astable2
    bcf    flags,OUTON    ;clear timer output on flag
    call   IsTgtZero      ;is target timer = 0?
    SEQ
    goto   seqEnd        ;if not, go to seqEnd
    movlw 1
    movwf  modest        ; otherwise, move
    call   MoveP1ToTgt    ; initialize target counter value
```

Electromechanical Timer Replacement

```
        goto    seqEnd        ;go to seqEnd

EASTable0
    bcf    flags,OUTON        ;clear timer output on flag
    call   MoveP1ToTgt        ;initialize target counter value
    btfsc  flags,INPON        ;is timer input on?
    incf   modest            ;if yes - move to mode state 1
    goto   seqEnd            ;go to seqEnd

EASTable1
    btfss  flags,INPON        ;is timer input still on?
    goto   EASTable1a        ;if not - go to EASTable1a
    bsf    flags,OUTON        ;set timer output on flag
    call   IsTgtZero          ;is target timer = 0?
    SEQ                    ;
    goto   seqEnd            ;if not, go to seqEnd
    incf   modest            ;otherwise, move to mode state 2
    call   MoveP2ToTgt        ;initialize target counter value
    goto   seqEnd            ;go to seqEnd

EASTable1a
    bcf    flags,OUTON        ;clear timer output on flag
    clrf   modest            ;go back to mode state 0
    goto   seqEnd            ;

EASTable2
    bcf    flags,OUTON        ;clear timer output on flag
    btfss  flags,INPON        ;is timer input still on?
    goto   EASTable2a        ;if not - go to EASTable2a
    call   IsTgtZero          ;is target timer = 0?
    SEQ                    ;
    goto   seqEnd            ;if not, go to seqEnd
    movlw  1                  ;otherwise, move
    movwf  modest            ; to mode state 0
    call   MoveP1ToTgt        ;initialize target counter value
    goto   seqEnd            ;go to seqEnd

EASTable2a
    clrf   modest            ;go back to mode state 0
    goto   seqEnd            ;go to seqEnd

Counter0
    bcf    flags,OUTON        ;clear timer output on flag
    call   MoveP1ToTgt        ;initialize target counter value
    incf   modest            ;move to state 1
    goto   seqEnd            ;go to seqEnd

Counter1
    bcf    flags,OUTON        ;clear timer output on flag
    btfsc  flags,RSTON        ;is reset input active?
    goto   Counter1a         ;yes it is - go to Counter1a
    btfss  flags,EDGON        ;have we detected an active input edge?
    goto   Counter1b         ;no we have not - go to Counter1b
    call   DecrTgt            ;decrement the target counter
    call   IsTgtZero          ;is target counter = 0?
    SEQ                    ;
    goto   Counter1b         ;nope - go to Counter1b
    incf   modest            ;move to mode state 2
    goto   seqEnd            ;go to seqEnd

Counter1a
    call   MoveP1ToTgt        ;re-initialize target counter value

Counter1b
    goto   seqEnd            ;go to seqEnd
```

Electromechanical Timer Replacement

Counter2

```
bsf    flags,OUTON    ;set timer output on flag
btfss  flags,RSTON    ;is reset input active?
goto   seqEnd         ;if not, go to seqEnd
call   MoveP1ToTgt    ;initialize target counter value
decf   modest        ;move back to mode state 1
goto   seqEnd         ;go to seqEnd
```

ACounter2

```
bsf    flags,OUTON    ;set timer output on flag
btfsc  flags,EDGON    ;have we detected an active input edge?
goto   ACounter2a     ;yes we have - go to ACounter2a
btfss  flags,RSTON    ;is reset input active?
goto   seqEnd         ;if not, go to seqEnd
call   MoveP1ToTgt    ;initialize target counter value
decf   modest        ;move back to mode state 1
goto   seqEnd         ;go to seqEnd
```

```
; auto reset handler (make sure configuration program does not allow terminal
; count value less than 2)
```

ACounter2a

```
bcf    flags,OUTON    ;clear timer output on flag
call   MoveP1ToTgt    ;initialize target counter
call   DecrTgt        ;decrement target counter
decf   modest        ;move back to mode state 1
goto   seqEnd         ;go to seqEnd
```

```
;*****
```

```
;* Start of program *
```

```
;*****
```

```
start   movlw    11001111b    ;disable wake-up on GPIO pin change, disable weak
                                   ; pullups, use internal clock for RTCC (positive edge),
                                   ; assign prescaler to WDT, set prescaler to 128
        option   ;
        movlw    GPTRIS       ;set GPIO
        tris     GPIO         ; TRIS register
        movf     RTCC,w       ;initialize
        movwf    oldRTCC      ; oldRTCC value
        call     InitRollOver ;initialize rollover counter
        clrf    tgtL         ;initialize
        clrf    tgtH         ; target count register
        movlw    DBNCTM       ;initialize
        movwf    dbncTmr     ; debounce timer
        clrf    inDbnc        ;set initial state of timer input debounce
                                   ; state machine
        clrf    rstDbnc      ;set initial state of reset input debouce
                                   ; state machine
        clrf    flags        ;clear flags register
        clrf    modest       ;clear timer mode state
```

```
;read the timer configuration from the EEPROM and copy it to the
```

```
;block of file registers used to hold the configuration data
```

readEE

```
movlw   6
movwf   bcount    ;set number of bytes to read as 6
movlw   mode      ;set FSR to point to 1st address into
movwf   FSR       ; configuration memory block
call    bStart    ;generate start bit
movlw   10100000b ;set slave address and write mode and
movwf   txbuf     ; copy into transmit buffer
call    tx        ; and send it
movlw   0         ;copy read start address
movwf   txbuf     ; into transmit buffer
call    tx        ; and send it
call    bStart    ;generate start bit
movlw   10100001b ;get slave address and read mode
movwf   txbuf     ;into transmit buffer
```

Electromechanical Timer Replacement

```

        call    tx            ;and transmit it
                                ;
rbyte   call    rx            ;read 1 byte from device
        movf   datai,w       ;copy byte into register
        movwf  INDO          ; pointed to by FSR
        incf   FSR           ;increment the FSR
        decfsz bcount        ;are all 6 bytes read?
        goto   lowack        ;no, send low ack and do another
        bsf    eeprom,do     ;yes, send high ack bit
        call   bitOut        ;to stop transmission
        call   bStop         ;and send a stop bit
        vgoto  infLoop       ;all done - go to the main program loop

lowack   bcf    eeprom,do     ;send low ack bit
        call   bitOut        ;to continue transmission
        goto   rbyte         ;and read another byte

;*****
;* Start of main program loop          *
;*****
infLoop

;handle the timer
HandleTimer
        movf   RTCC,w        ;get the current RTCC counter value
        movwf  temp          ; and temporarily save it
        subwf  oldRTCC,w     ;has the RTCC
        SLT                    ; rolled over?
        goto   decRoll       ;if yes - decrement the rollover counter
        goto   tmrEnd        ;else goto tmrEnd

decRoll
        decf   dbncTmr       ;decrement the debounce timer
        movlw  1              ;subtract 1
        subwf  rollL         ; from rollL (low byte)
        SGE                    ;skip if borrow did not occur
        subwf  rollH         ;otherwise, subtract 1 from rollH (high byte)
        movlw  0              ;is rollover
        iorwf  rollL,w       ; counter
        SNE                    ; =
        iorwf  rollH,w       ; 0?
        SEQ                    ;
        goto   tmrEnd        ;if no - goto tmrEnd
        call   InitRollOver  ;re-initialize rollover counter

;handle .1 second target counter if not in an event counter mode
        movlw  COUNTR        ;are we in an
        subwf  mode,w        ; event counter
        SLT                    ; mode?
        goto   tmrEnd        ;if yes - goto tmrEnd
        call   IsTgtZero     ;otherwise, is target counter = 0?
        SEQ                    ;if it is, skip
        call   DecrTgt       ;otherwise, decrement the target counter

tmrEnd
        movf   temp,w        ;update old RTCC value
        movwf  oldRTCC      ; from temp register

;handle inputs
handleIn
        bcf    flags,EDGON   ;reset the input edge flag
; handle debouncing of the inputs
        clrw   dbncTmr,w     ;is debounce timer
        iorwf  dbncTmr,w     ; = 0?
        SEQ                    ;if yes, skip next instruction
        goto   doSequencer   ;otherwise, go to doSequencer
```

Electromechanical Timer Replacement

```
; handle TimerIn input debounce state machine
    call    doTmrInState ;run timer input debounce state machine
    movlw  2             ;is current state
    subwf  inDbnc,w     ; >= 2?
    SGE                    ;if so, skip next instruction
    goto   TInEnd3     ;goto TInEnd3 (input low)

;input high
    btfsc  flags,INPHI  ;was timer input previously low?
    goto   TInEnd1     ;if not - go to TInEnd1
    btfsc  tmropt,TRLEDG ;are we configured for rising edge detect?
    goto   TInEnd1     ;if not - go to TInEnd1
    bsf    flags,EDGON  ;set the edge detect flag

TInEnd1  bsf    flags,INPHI ;set timer input high flag
    btfsc  tmropt,LOWIN ;is input configured to be active low?
    goto   TInEnd2     ;if yes - go to TInEnd2
    bsf    flags,INPON  ;if no - set timer input active flag
    goto   doRstIn     ;go to doRstIn
TInEnd2  bcf    flags,INPON ;clear timer input active flag
    goto   doRstIn     ;go to doRstIn

;input low
TInEnd3  btfss  flags,INPHI ;was timer input previously high?
    goto   TInEnd4     ;if not - go to TInEnd3
    btfss  tmropt,TRLEDG ;are we configured for trailing edge detect?
    goto   TInEnd4     ;if not - go to TInEnd3
    bsf    flags,EDGON  ;set the edge detect flag
TInEnd4  bcf    flags,INPHI ;clear timer input high flag
    btfss  tmropt,LOWIN ;is input configured to be active low?
    goto   TInEnd5     ;if no - go to TInEnd5
    bsf    flags,INPON  ;if yes - set timer input active flag
    goto   doRstIn     ;go to doRstIn
TInEnd5  bcf    flags,INPON ;clear timer input active flag

;handle TimerReset input
doRstIn
    call    doRstInState ;run reset input state machine
    movlw  2             ;is current state
    subwf  rstDbnc,w     ; >= 2?
    SGE                    ;if so, skip next instruction
    goto   RstEnd2     ;if no - go to RstEnd2

;reset input high
    bsf    flags,RSTHI  ;clear Timer Reset high flag
    btfsc  tmropt,LOWRST ;is reset input configured to be active low?
    goto   RstEnd1     ;if yes - go to RstEnd1
    bsf    flags,RSTON  ;set reset input on flag
    goto   dbncEnd     ;go to dbncEnd
RstEnd1  bcf    flags,RSTON ;clear reset input on flag
    goto   dbncEnd     ;go to dbncEnd

;reset input low
RstEnd2  bcf    flags,RSTHI ;clear Timer Reset high flag
    btfss  tmropt,LOWRST ;is reset input configured to be active low?
    goto   RstEnd3     ;if no - go to RstEnd3
    bsf    flags,RSTON  ;set reset input on flag
    goto   dbncEnd     ;go to dbncEnd
RstEnd3  bcf    flags,RSTON ;clear reset input on flag

dbncEnd  movlw  DBNCTM   ;reset
    movwf  dbncTmr     ; debounce timer

;handle sequencer state
doSequencer
```

Electromechanical Timer Replacement

```
        call    doSeqState    ;run mode sequencer state machine
        btfss  flags,OUTON   ;is timer output on?
        goto   outOff        ;nope - go to outOff

; timer output on
outOn
        btfsc  tmropt,LOWOUT ;is timer output configured to be active low?
        goto   outOn1       ;if yes - go to outOn1
        bsf   GPIO,TMROUT   ;set timer output high
        goto   outEnd        ;go to dbncEnd
outOn1  bcf   GPIO,TMROUT   ;set timer output low
        goto   outEnd        ;go to outEnd

; timer outout off
outOff  btfss  tmropt,LOWOUT ;is timer output configured to be active low?
        goto   outOff1      ;if no - go to outOff1
        bsf   GPIO,TMROUT   ;set timer output high
        goto   outEnd        ;go to dbncEnd
outOff1 bcf   GPIO,TMROUT   ;set timer output low

outEnd
        clrwdt                ;reset the watchdog timer
        goto   infLoop        ;do it again! and again! and again!

;*****
;* Reset vector *
;*****
; For 12C508, this location contains movlw XX, where XX is the calibration value
; for the on-board oscillator - thus the real reset vector is at address 0
        org    0x1fff        ;location of "reset" vector

;*****
;* End of program *
;*****
        end
```



Electromechanical Timer Replacement

Time Delay Relay Family

*Author: Paul McCoy
Zykora
Waukesha, WI
USA
email: paul_mccoy@zykora.com*

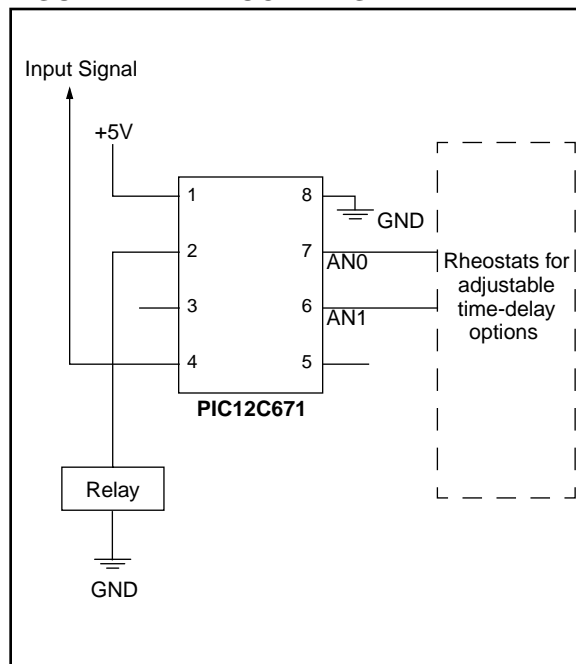
APPLICATION OPERATION

This design entry uses the PICmicro's ability to be programmed in-circuit to create a large family of devices from one simple design. This will allow the cost savings of mass production while giving the sales leverage of customized components without long turn-around times or increased inventory requirements. Additionally, since the target market for this product is consumer and industrial equipment manufacturers, the agency approvals (UL, CSA, European, etc.) should be easier to get and maintain since there is only a few hardware prototypes for the many incarnations.

The devices that are being replaced either provide a time delay between the control signal coming on and the relay activating, or provide a delay between the control signal going off and the relay deactivating. The time delay is either a set value at manufacture or adjustable within a set range. This functionality is duplicated in this design using the PIC12C671, a standard 5 VDC relay, and (in the case of an adjustable delay) a potentiometer. In fact this design makes it possible to have separate delays for turn on and turn off.

As mentioned previously, there are many implementations of the software for one common hardware design. The source code included in this application note is based on the dual adjustable model since the other implementations are a subset of this one. The range for the adjustment is 0 – 31 seconds in this example.

FIGURE 1: BLOCK DIAGRAM



MICROCHIP TOOLS USED

Assembler/Compiler Version:

MPLAB version 3.22.0 (MPASM)

Microchip Technology Incorporated, has been granted a non-exclusive, worldwide license to reproduce, publish and distribute all submitted materials, in either original or edited form. The author has affirmed that this work is an original, unpublished work and that he/she owns all rights to such work. All property rights, such as patents, copyrights and trademarks remain with author.

Electromechanical Timer Replacement

APPENDIX A: SOURCE CODE

```
;*****
;
;           TM_DLY01.ASM
;           Variable time delay, on & off, relay program
;*****
;           AUTHOR:           Paul McCoy
;           DATE:            31 May 1997
;           ORGANIZATION:    Zykora Inc.
;           REVISION:        1.00.00
;
;*****
;
;           LIST    p=12C671; PIC12C671 is the target processor
#include <P12C671.INC>

CBLOCK      0x20                ; autosets register variable addresses
            flag                ; flag register for program flags
            second_cnt         ; counter for one second timer
            temp_stat          ; holds status register for interrupts
            temp_w             ; holds W register for interrupts
            time_cnt

ENDC

;*****
;
;           FLAG BITS
;*****

COMMAND     equ      00        ; Set - input line is set
ACTUAL      equ      01        ; Set - output line is set
TIMING      equ      02        ; Set - Timer is running

;*****
;
;           OTHER DEFINES
;*****

LAST_LOCATE equ      0x3ff

;*****
;
;           Program
;*****

org         0x00                ;reset vector
goto       start

            org         0x04                ;interrupt vector
            movwf      temp_w                ;note: if Bank1 is used this space must
            ;be reserved

            swapf      STATUS,W
            bcf        STATUS,RP0
            movwf      temp_stat
            btfsc     INTCON,2
            call       Timer
            btfsc     INTCON,1
            bcf        INTCON,1
            btfsc     INTCON,0
            call       Input_change
            swapf      temp_stat,W
            movwf     STATUS
            swapf      temp_w,F
            swapf      temp_w,W
            retfie

;*****
;
;           Initialization block
```

Electromechanical Timer Replacement

```
*****
start:
    clrf        GPIO
    clrf        flag
    bsf         STATUS,RP0        ;switch to Bank 1
    movlw       B'11000101'
    movwf      OPTION_REG        ;tmr0 scaled 64:1
    movlw       B'00000100'      ;GPIO 0 & 1 A/D; 3, 4, & 5 digital
    movwf      ADCON1
    movlw       B'11101111'      ;GPIO 5 output; all others input
    movwf      TRISIO
    clrf        PIE1
    call        LAST_LOCATE
    movwf      OSCCAL
    bcf         STATUS,RP0        ;switch back to Bank 0
    movlw       B'01000001'      ;select channel 0 and turn on A/D
    movwf      ADCON0
    btfsc      GPIO,3
    call        Initial_set
    movlw       -D'125'          ;count up 125 to rollover
    movwf      TMR0
    movlw       B'10101000'      ;enable timer interrupt
    movwf      INTCON

;*****
;                               Main program loop
;*****

top:
    goto       top

;*****
;                               Normally energized initialization routine
;*****

Initial_set:
    bsf        GPIO,3
    bsf        flag,ACTUAL
    bsf        flag,COMMAND
    bsf        ADCON0,3
    return

;*****
;                               A/D Conversion Subroutine
;                               This subroutine assumes the A/D channel has already been
;                               selected and stabilized and that the output destination is
;                               specified in FSR. It waits for the A/D to be complete before
;                               returning, not using the A/D interrupt.
;*****

Adloop:
    bsf        ADCON0,2

adwait:
    btfss     PIR1,6
    goto      adwait
    bcf       PIR1,6
    return

;*****
;                               Timer interrupt routine
;*****
```

Electromechanical Timer Replacement

```
Timer:
    bcf          INTCON,2
    movlw       -D'125'          ;count up 125 to rollover
    movwf      TMR0
    decfsz     second_cnt,F
    return
    movlw      D'125'          ;125*125*64=1 million clocks
                                ;equals one second
    movwf     second_cnt
    decfsz    time_cnt,F
    return
    btfss     flag,TIMING
    return
    bcf       flag,TIMING
    btfsc     flag,COMMAND
    goto      set_out
    bcf       GPIO,5
    bcf       flag,ACTUAL
    return

set_out:
    bsf       GPIO,5
    bsf       flag,ACTUAL
    return
```

```
*****
;
;          Input interrupt routine
*****
```

```
Input_change:
    bcf          INTCON,0
    btfss       GPIO,3
    goto        input_off
    btfsc       flag,COMMAND
    return
    bsf         flag,COMMAND
    btfsc       flag,ACTUAL
    goto        error_on
    call        Adloop
    bsf         ADCON0,3
    rrf         ADRES,F
    rrf         ADRES,F
    rrf         ADRES,F
    movf        ADRES,W
    btfsc       STATUS,Z
    goto        timed_on
    movwf      time_cnt
    movlw      -D'125'
    movwf      TMR0
    bcf         INTCON,2
    movlw      D'125'
    movwf      second_cnt
    bsf         flag,TIMING
    return

error_on:
    bcf         flag,TIMING
    return

timed_on:
    bsf         GPIO,5
    bsf         flag,ACTUAL
    return

input_off:
    btfss       flag,COMMAND
    return
    btfss       flag,ACTUAL
```

Electromechanical Timer Replacement

```
        call      Adloop
        bcf       ADCON0,3
        rrf      ADRES,F
        rrf      ADRES,F
        rrf      ADRES,F
        movf     ADRES,W
        btfsc    STATUS,Z
        goto     timed_off
        movwf    time_cnt
        movlw    -D'125'
        movwf    TMR0
        bcf      INTCON,2
        movlw    D'125'
        movwf    second_cnt
        bsf      flag,TIMING
        return

error_off:
        bcf      flag,TIMING
        return

timed_off:
        bcf      GPIO,5
        bcf      flag,ACTUAL
;
        END
```



Electromechanical Timer Replacement

Timer Controllers

*Author: Bret Walters
Inter.tec.
Ontario, Canada
email: bretw@ibm.net*

SUGGESTED APPLICATIONS

Egg Timer

This device could be used in many ways, to solve the "egg timer" example; it could be battery operated and have one or two buttons. The version shown has one button, its functions are stop/clear and start/reset. To operate, the button is pressed once for each timer increment. The timer automatically starts counting when the button is no longer pressed. Pressing (and perhaps holding) the button again, will stop and reset the timer. Two buttons could easily be used but are unnecessary. For this operation, I suggest an external crystal or RC oscillator to give a slow clock that is then divided. This reduces code complexity by not needing as many counting bits and reduces power consumption. The circuit could be powered from a lithium 3V cell.

Garden Watering Controller

The device would be powered by the AC outlet in this example and drive a opto-isolated triac to produce output, a relay or other device could also be used but a triac is recommended. To drive a DC load a nice power MOSFET will do fine. Observe the schematic to see how the PICmicro™ is powered from the AC line.

Since the PICmicro needs little current, the current is taken from a diode which is connected to a simple voltage divider with a low power 5.1V zener diode and a resistor (5W recommended to prevent overheating due to power dissipation). The output ripple is removed with a small capacitor, sat 1 μ F to produce stable power.

Conveniently enough, this now gives a source of 60 Hz accurate clocking signal that can be counted and used for accurate timing. **Trimming the clock is no longer necessary!** The timer simply debounces and counts these pulses and compares it to the desired value. The desired value is set using the same one-button or two button interface as the Egg-timer. The device could likely be built small enough to fit in an adapter socket or power bar.

Freezer Life Extender/Motor Life Extender/ Power-on Delay Circuit

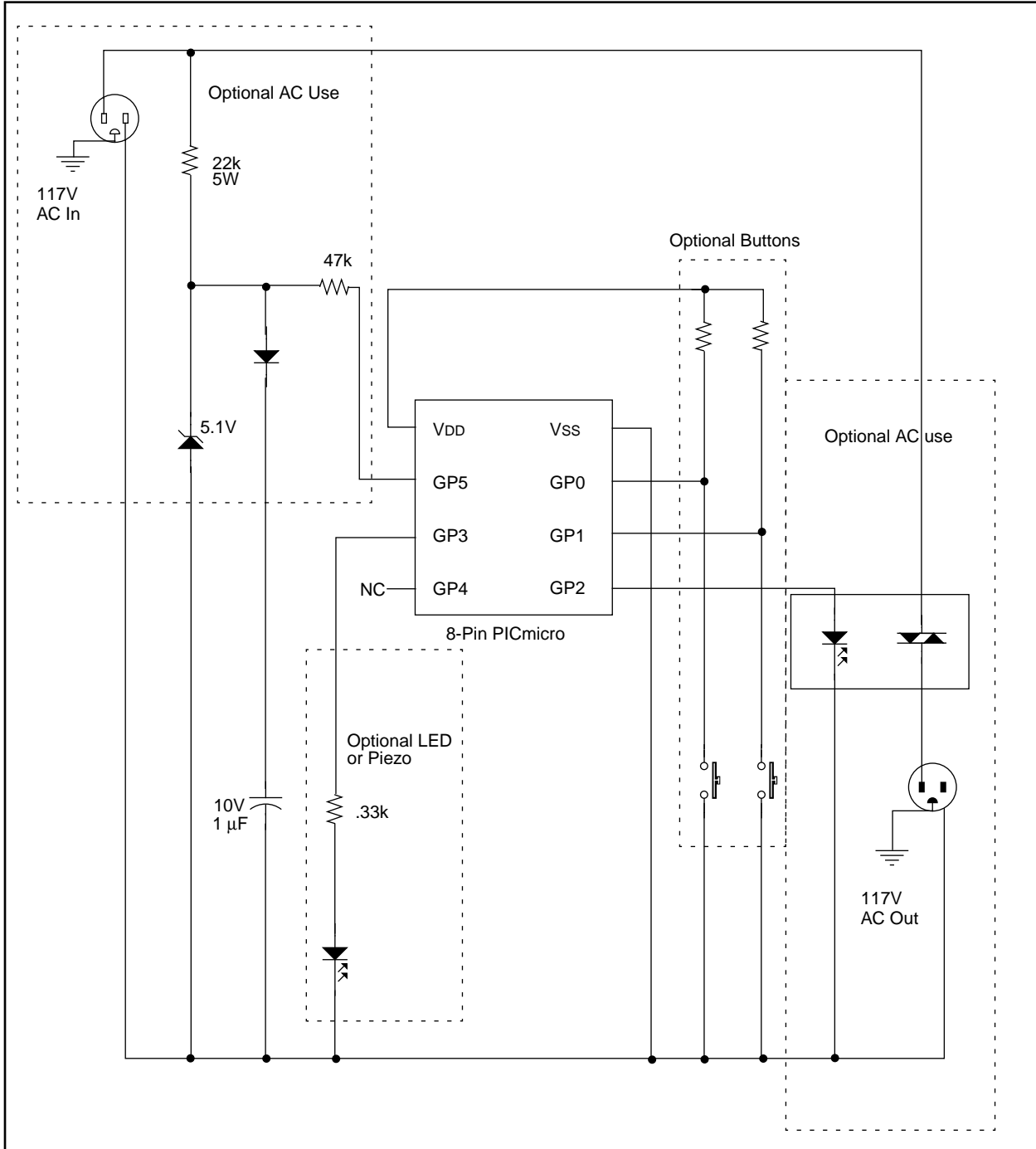
After a power failure, the compressor of a freezer becomes hard to drive due to the settling of the refrigerant. This causes a high load on the AC compressor motor and slowly reduces its life span. Furthermore, the load on the AC line is quite high and if other motors are on the same circuit, breakers can trip.

This could be easily averted by building a Garden Watering Controller into the freezer startup and having no user buttons. The delay would be set to approximately. 5 minutes. This delay gives the freezer time to slowly move the refrigerant, greatly reducing load. Plus, the freezer no longer places a load on the AC line at the same time as all other devices that come back on line at the end of a power failure.

Microchip Technology Incorporated, has been granted a non-exclusive, worldwide license to reproduce, publish and distribute all submitted materials, in either original or edited form. The author has affirmed that this work is an original, unpublished work and that he/she owns all rights to such work. All property rights, such as patents, copyrights and trademarks remain with author.

Electromechanical Timer Replacement

FIGURE 1: SCHEMATIC TIMER CONTROLLER (EGG, LAWN, FREEZER, ETC.)



Electromechanical Timer Replacement

CODE NEEDED

- Debounce input switches and/or 60 Hz input if needed
- Count input pulses or the clock
- Divide the prescaler for the clock by 256 (if clock is 4 MHz and the 60 Hz method is not used)
- Drive an LED or buzzer, depending on application
- Read input ports to determine functions and change the memory appropriately to store values
- Compare memory with memory (to determine if time is up or not)
- Provide feedback to user (beep or flash as button(s) are pressed appropriately if needed)

CLOCK CYCLES

This circuit should easily have enough clock cycles to get the job done. The intended oscillator used is the 4 MHz internal clock. The circuit could easily use any oscillator, but this is likely unnecessary and is an added cost. However, if low power operation is desired, try using the external oscillator configuration.