```
1
2   ;
3   ;
4           LIST            P=PIC16F628A
5           ERRORLEVEL      -302    ;SUPPRESS BANK SELECTION MESSAGES
6           __CONFIG        03F10H  ;see below
7   ;
8   ;       bit:   13 12 11 10 9 8 7 6 5 4 3 2 1 0
9   ;     value:    1  1  1  1 1 1 0 0 0 1 0 0 0 0
10  ;
11  ; bit 13 = 1   code protection off
12  ; bit 12-9 = 1 not u sed
13  ; bit 8  = 1   data memory protyection off
14  ; bit 7  = 0   normal programming
15  ; bit 6  = 0   no brown out reset
16  ; bit 5  = 0   reset pin is digital IO
17  ; bit 3  = 0   power up timer enable
18  ; bit 2  = 0   watch dog timer disabled
19  ; bits 4,1,0 = 100 int oscillator (4 MHz) pins are IOs
20  ;
21  ; IO port designation:
22  ;
23  ;       bit:     7     6     5     4     3     2     1     0
24  ;     PORTA:     x     x     x   100Hz   x    out   out   out
25  ;                                inp          R     G     B
26  ;     TRISA:     1     1     1     1     1     0     0     0    = F8
27  ;
28  ;
29  ;       bit:     7     6     5     4     3     2     1     0
30  ;     PORTB:     x     x     x  manual meter meter meter   x
31  ;                              input    1     2     0
32  ;     TRISB:     1     1     1     1     0     0     0     1    = F1
33  ;
34  INCLUDE P16F628A.INC
35  ;
36  ;timer0 is configured to generate an interrupt every 200us
37  ;At 4 MHz oscillator frequency, the internal clock runs at 1 mHz = 1 us
38  ;the prescaler is set at 1:4 so that counting period of timer0 is 4us
39  ;this requires 200/4=50 counts. This makes the reload value for
40  ;TMR0=256-50=206 or 0CEH
41  ;
42  ;
43  ;THE CONSTANTS
44  NCYCLE_REL      EQU     006H    ;number of cycles making up one transistion step
45                                  ;this in principle determines the time every transistion
46                                  ;takes. 01H is highest speed. Increase for slower transistions.
47  NBURST_MAX      EQU     01FH    ;number of bursts in a cycle, and steps in a transistion
48  FILTREL         EQU     02CH    ;100HZ filter counter reload
49  KEYREL          EQU     020H    ;reload value for the key debounce counter
50  MAXHRS          EQU     00CH    ;number of hours in a day 18H (=24dec) or 0CH (=12)
51  MAINS           EQU     032H    ;mains frequency
52  ;
53  ;
54  ;The Bits
55  AB              EQU     0       ;if 1 color A is on
56  FF              EQU     1       ;reflects 100Hz input
57  SYNK            EQU     2       ;set on a 0->1 transition on the 100Hz input
58  NEW100          EQU     3       ;new 100Hz period found
59  NEWSEC          EQU     4       ;new second found flag
60  GOTIME          EQU     5       ;if 1 clock is running
61  NEWKEY          EQU     6       ;if 1 key was pressed
62  ;
63  ;
64  ;The Bytes
65  BITS1           EQU     020H    ;..,NEWKEY,GOTIME,NEWSEC,NEW50,SYNK,FF,AB
66  BITS2           EQU     021H    ;..,..,..,..,..,..,..,..
67  W_TEMP          EQU     022H    ;save for W
68  STATUS_TEMP     EQU     023H    ;save for STATUS
69  PCLATH_TEMP     EQU     024H    ;save for PCLATH
70  COLOR           EQU     025H    ;current color code
71  LCNT            EQU     026H    ;level counter
```

```
LTMP            EQU     027H    ;temp for the level routine
COLA            EQU     028H    ;color A
COLB            EQU     029H    ;color B
A2B_RAT         EQU     02AH    ;ratio between COLB and COLA (00=colB)
NBURST          EQU     02BH    ;burst counter (number of burst in a cycle)
NCYCLE          EQU     02CH    ;cycle counter (number of cycles in a transition step)
INDX            EQU     02DH    ;pointer in the color table
FILT100         EQU     02EH    ;100Hz filter counter
CNT100          EQU     02FH    ;counts down 100 100HZ periods for a second
SEC             EQU     030H    ;seconds
MIN             EQU     031H    ;minutes
HRS             EQU     032H    ;hours
SECPWM          EQU     033H    ;pwm counter for seconds
MINPWM          EQU     034H    ;pwm counter for minutes
HRSPWM          EQU     035H    ;pwm counter for hours
KEYCNT          EQU     036H    ;input key debounce counter
;
;
                org             0000H
                goto            init
;
                org             0004H
                goto            tmrint
;
; initialize special function registers (machine parameters)
init:           bcf     STATUS,RP1      ;goto bank 0
                bcf     STATUS,RP0      ;
                movlw   0x07            ;all pins digital IO
                movwf   CMCON           ;
                movlw   000H            ;init value PORTA
                movwf   PORTA           ;
                movlw   000H            ;init value PORTB
                movwf   PORTB           ;
                movlw   0CE             ;init TMR0
                movwf   TMR0            ;
                movlw   03H             ;PCLATH to upper page
                movwf   PCLATH
                bsf     STATUS,RP0      ;goto bank 1
                movlw   0F8H            ;init PORTA
                movwf   TRISA           ;
                movlw   0F1H            ;init PORTB
                movwf   TRISB           ;
                movlw   0D1H            ;timer0 on internal clock, presc 1:4
                movwf   OPTION_REG      ;
                bcf     STATUS,RP0      ;return to bank 0
                movlw   0A0H            ;timer0 interrupt on
                movwf   INTCON          ;
;
; initialize program variables and bits
                movlw   03FH            ;initialize level counter
                movwf   LCNT            ;
                clrf    NBURST          ;
                bcf     BITS1,AB        ;
                movlw   NCYCLE_REL      ;
                movwf   NCYCLE          ;
                clrf    A2B_RAT         ;
                movlw   00H             ;pointers to first colors in table
                call    c_table         ;
                movwf   COLA            ;
                movlw   01H             ;
                call    c_table         ;
                movwf   COLB            ;
                movlw   02H             ;
                movwf   INDX            ;
                movlw   FILTREL         ;initialize the 100Hz filter
                movwf   FILT100         ;
                movlw   064H            ;a 100 100Hz periods in one second
                movwf   CNT100          ;
                bcf     BITS1,GOTIME    ;halt clock
                clrf    SEC             ;set time to 00:00:00
                clrf    MIN             ;
```

```
                clrf    HRS             ;
                movlw   03CH            ;initialize the pwm counter for seconds to 60
                movwf   SECPWM          ;
                movlw   03CH            ;initialize the pwm counter for minutes to 60
                movwf   MINPWM          ;
                movlw   018H            ;initialize the pwm counter for hours to 24
                movwf   HRSPWM          ;
                movlw   KEYREL          ;initialize KEYCNT with KEYREL
                movwf   KEYCNT          ;
                bcf     BITS1,NEWKEY    ;reset NEWKEY flag
;
;
;
main:           btfsc   PORTB,4         ;skip if input key is pressed at start-up (test program)
                goto    main_6          ;othrwise normal clock routine
main_1:         btfss   PORTB,4         ;wait until key is released again
                goto    main_1          ;               '
;
;test program
main_2:         bcf     BITS1,NEWKEY    ;
                clrf    SEC             ;sec,min,hrs 00:00:00 (minimum scale)
                clrf    MIN             ;
                clrf    HRS             ;
main_3:         btfss   BITS1,NEWKEY    ;wait until key pressed
                goto    main_3          ;
                bcf     BITS1,NEWKEY    ;
                movlw   01EH            ;sec:min:hrs 12:30:30 (half scale)
                movwf   SEC             ;
                movwf   MIN             ;
                movlw   MAXHRS/2        ;
                movwf   HRS             ;
main_4:         btfss   BITS1,NEWKEY    ;wait until key pressed
                goto    main_4          ;
                bcf     BITS1,NEWKEY    ;
                movlw   03CH            ;sec:min:hrs 24:60:60 (full scale)
                movwf   SEC             ;
                movwf   MIN             ;
                movlw   MAXHRS          ;
                movwf   HRS             ;
main_5:         btfss   BITS1,NEWKEY    ;wait until key pressed
                goto    main_5          ;
                goto    main_2          ;loop test program
;
;set hours
main_6:         btfss   BITS1,NEWKEY    ;wait until key is pressed
                goto    main_6          ;
                bcf     BITS1,NEWSEC    ;
                bcf     BITS1,NEWKEY    ;key was pressed start adjusting hours
                movlw   064H            ;start with a full new second
                movwf   CNT100          ;
main_7:         btfsc   BITS1,NEWKEY    ;
                goto    main_8          ;jump if key was pressed
                btfss   BITS1,NEWSEC    ;skip if new second
                goto    main_7          ;loop if no key and no new second
                bcf     BITS1,NEWSEC    ;
                incf    HRS,F           ;increment HRS until HRS=24
                movf    HRS,W           ;
                xorlw   MAXHRS          ;
                btfss   STATUS,Z        ;
                goto    main_7          ;loop if not yet 24
                clrf    HRS             ;reset HRS otherwise
                goto    main_7          ;loop
main_8:         bcf     BITS1,NEWKEY    ;first reset newkey flag
;
;set minutes
main_9:         btfss   BITS1,NEWKEY    ;wait until key is pressed
                goto    main_9          ;
                bcf     BITS1,NEWSEC    ;
                bcf     BITS1,NEWKEY    ;key was pressed start adjusting minutes
                movlw   064H            ;start with a full new second
                movwf   CNT100          ;
```

```
main_10:        btfsc   BITS1,NEWKEY    ;
                goto    main_11         ;jump if key was pressed
                btfss   BITS1,NEWSEC    ;skip if new second
                goto    main_10         ;loop if no key and no new second
                bcf     BITS1,NEWSEC    ;
                incf    MIN,F           ;increment MIN until MIN=60
                movf    MIN,W           ;
                xorlw   03CH            ;
                btfss   STATUS,Z        ;
                goto    main_10         ;loop if not yet 60
                clrf    MIN             ;reset MIN otherwise
                goto    main_10         ;loop
main_11:        bcf     BITS1,NEWKEY    ;first reset newkey flag
;
;wait for starting of the clock
main_12:        btfss   BITS1,NEWKEY    ;wait until key is pressed
                goto    main_12         ;
                bcf     BITS1,NEWSEC    ;
                bcf     BITS1,NEWKEY    ;key was pressed
                movlw   064H            ;start with a full new second
                movwf   CNT100          ;
                bsf     BITS1,GOTIME    ;start the clock
;
;loop forever or until key is pressed, then goto set time again.
main_13:        btfss   BITS1,NEWKEY    ;wait until key is pressed
                goto    main_13         ;
                bcf     BITS1,NEWKEY    ;if key is pressed set time again
                goto    main_6          ;
;
;
;
;subsroutine: tmrint (called every 200us)
;This routine is called every 200us when an roll-over of timer0 occurs.
;after resetting of timer0 a number of "poll" subroutines are called
;which take care of the ambilight feature, filtering of the mains frequency,
;time keeping and debouncing of the inout key
tmrint:         call    push            ;save variables
                bcf     INTCON,T0IF     ;reset interrupt flag
                movlw   0D3H            ;reload timer0
                movwf   TMR0            ;
                movlw   03H             ;pclath to page 3
                movwf   PCLATH          ;
                call    rnd_col         ;process the ambi-light feature
                call    flipflop        ;get the 100Hz input frequency
                call    filter          ;filter it
                call    second          ;count down the number of periods for ome second
                call    time            ;keep the time
                call    pwm             ;process the pulse-width modulation for the analog meter
outputs
                call    inkey           ;get and debounce the input key
                call    pop             ;
                retfie
;
;
;subroutine: flipflop (called every 200us from "tmrint")
;This routine reads the 100Hz inout and copies it into "FF"
;When a 0->1 transition is found (beginning of a new period)
;flag "SYNK" is set.
flipflop:       btfsc   BITS1,FF        ;skip if FF=0
                goto    flipflop_1      ;jump if FF=1
                btfss   PORTA,4         ;only if FF=0 and PORTA,5=1 skip
                return                  ;otherwise return
                bsf     BITS1,FF        ;FF:=1
                bsf     BITS1,SYNK      ;SYNK:=1
                return                  ;
flipflop_1:     btfsc   PORTA,4         ;only if FF=1 and PORTA,5=0 skip
                return                  ;otherwise return
                bcf     BITS1,FF        ;FF:=0
                return                  ;
;
;
```

```
;subroutine: filter (called every 200us from routine "tmrint")
;This routine filters the 100Hz input signal from transients. The routine is
;called every 200us. Every call the routine decrements counter FILT50 until
;the counter reaches zero. This takes about 90% of the 50Hz period. Next the
;routine waits for a 0 to 1 transition (SYNK=1). At that point
;a new 100Hz period is found, and counter FILT100 is reloaded and SYNK is reset.
filter:        movf    FILT100,F      ;if FILT100 not 0 yet skip and decrement
               btfsc   STATUS,Z       ;
               goto    filter_1       ;if not 0 jump
               decf    FILT100,F      ;decrement FILT100
               return                 ;
filter_1:      btfss   BITS1,SYNK     ;if FILT100=0 and SYNK, new 100Hz period found
               return                 ;
               bcf     BITS1,SYNK     ;clear SYNK flag
               bsf     BITS1,NEW100   ;new filtered 100Hz period found
               movlw   FILTREL        ;reload filter
               movwf   FILT100        ;
               return
;
;
;subroutine: second (called every 200us from routine "tmrint")
;This routine counts down 100 NEW100 flags for one second. When a new
;second is reached, the NEWSEC flag is set.
second:        btfss   BITS1,NEW100   ;skip if NEW100 flag is set
               return                 ;
               bcf     BITS1,NEW100   ;clear NEW100 flag
               decfsz  CNT100,F       ;CNT100 := CNT100-1
               return                 ;return if not yet zero
               bsf     BITS1,NEWSEC   ;if zero new second found
               movlw   MAINS*2        ;reload mains frequency counter
               movwf   CNT100         ;
               return
;
;
;subroutine: time (called every 200us from routine "tmrint")
;This is the actual time keeping routine. The time is kept in 6 bytes:
;SEC1 : seconds units,  SEC10: seconds tens
;MIN1 : minutes units,  MIN10: minutes tens
;HRS1 : hours units,    HRS10: hours units
;The routine first checks if the GOTIME flag is set. When set the routine
;checks the NEWSEC flag. When it is set, the time is updated.
time:          btfss   BITS1,GOTIME   ;is the clock ticking ?
               return                 ;
               btfss   BITS1,NEWSEC   ;is there a new second ?
               return                 ;
               bcf     BITS1,NEWSEC   ;clear the new second
               incf    SEC,F          ;increment SEC until SEC=60
               movf    SEC,W          ;
               xorlw   03CH           ;
               btfss   STATUS,Z       ;
               return                 ;
               clrf    SEC            ;reset SEC
               incf    MIN,F          ;increment MIN until MIN=60
               movf    MIN,W          ;
               xorlw   03CH           ;
               btfss   STATUS,Z       ;
               return                 ;
               clrf    MIN            ;reset MIN
               incf    HRS,F          ;increment HRS until HRS=24
               movf    HRS,W          ;
               xorlw   MAXHRS         ;
               btfss   STATUS,Z       ;
               return                 ;
               clrf    HRS            ;
               return                 ;
;
;
;subroutine: pwm (called every 200us from routine "tmrint")
;This peace of program generates the pulse width modulated output which drives the analog
meters.
;The program consists fo four almost identical subparts: on for the seconds, one for the minutes
```

```
;and one for the hours. Lets for example look at the last loop which is for the hours.
;This loop counts down counter HRSPWM from 24 to 0 every call. When the counter reaches 0 it
resets
;output and reloads the counter. before returning the counter value is compared to HRS. When
they are
;equal the oputput is set.
pwm:            decfsz  SECPWM,F        ;sec pwm counter - 1
                goto    pwm_1           ;return if not yet 0
                movlw   03CH            ;reload pwm counter with 60dec and clear sec output M0
                movwf   SECPWM          ;
                bcf     PORTB,1         ;
pwm_1:          movf    SECPWM,w        ;set output M0 if pwm counter = sec
                xorwf   SEC,w           ;
                btfsc   STATUS,Z        ;
                bsf     PORTB,1         ;
                decfsz  MINPWM,F        ;min pwm counter - 1
                goto    pwm_2           ;return if not yet 0
                movlw   03CH            ;reload pwm counter with 60dec and clear min output M1
                movwf   MINPWM          ;
                bcf     PORTB,2         ;
pwm_2:          movf    MINPWM,w        ;set output M1 if pwm counter = min
                xorwf   MIN,w           ;
                btfsc   STATUS,Z        ;
                bsf     PORTB,2         ;
                decfsz  HRSPWM,F        ;hrs pwm counter - 1
                goto    pwm_3           ;return if not yet 0
                movlw   MAXHRS          ;reload pwm counter with 24dec or 12dec
                movwf   HRSPWM          ;
                bcf     PORTB,3         ;
pwm_3:          movf    HRSPWM,w        ;set output M2 if pwm counter = hrs
                xorwf   HRS,w           ;
                btfsc   STATUS,Z        ;
                bsf     PORTB,3         ;
                return
;
;
;subroutine: inkey (called every 200us from routine "tmrint")
;This routine processes and debounces the push-button key.
;When the key is not pressed, counter KEYCNT is initialized to KEYREL.
;Once the key is pressed, KEYCNT is counted down every call. When
;KEYCNT reaches 0, the NEWKEY flag is set. KEYCNT remains 0
;to indicate that a valid key pressed was found
inkey:          btfsc   PORTB,4         ;skip if key is pressed, if not reload counter
                goto    inkey_1         ;
                movf    KEYCNT,W        ;if KEYCNT=0 return is else continue
                btfsc   STATUS,Z        ;
                return                  ;
                decfsz  KEYCNT,F        ;KEYCNT:=KEYCNT-1 is zero skip,else return
                return                  ;
                bsf     BITS1,NEWKEY    ;NEWKEY found
                return                  ;
inkey_1:        movlw   KEYREL          ;reload KEYCNT with KEYREL
                movwf   KEYCNT          ;
                return
;
;
;subroutine: push
;This routine saves W,STATUS and PCLATH during an interrupt
push:           movwf   W_TEMP          ;copy w to temp register
                swapf   STATUS,W        ;swap status to be saved in W
                clrf    STATUS          ;bank 0
                movwf   STATUS_TEMP     ;save status register
                movf    PCLATH,W        ;only required if using pages 1, 2, 3
                movwf   PCLATH_TEMP     ;save pclath
                clrf    PCLATH          ;page zero
                return                  ;return
;
;
;subsroutine: pop
;This routine recalls W, STATUS and PCLATH when the interrupt service routine
;is completed.
```

```
pop:            movf    PCLATH_TEMP,W  ;restore PCLATH
                movwf   PCLATH         ;move W to PCLATH
                swapf   STATUS_TEMP,W  ;swap STATUS_TEMP register into W
                movwf   STATUS         ;move W into STATUS register
                swapf   W_TEMP,F       ;swap W_TEMP
                swapf   W_TEMP,W       ;swap W_TEMP into W
                return
;
;
;subroutine: rnd_col (called every 200us from routine "tmrint")
;This single subroutine displays a sequence of colors from a table on a LED.
;The routine generates a smooth transistion between the colors. The mixing
;of the colors, in the program called COLA and COLB, is done by multiplexing
;between the two colors. The initial duty-cycle in this switching is
;such that only COLA is shown. The duty-cycle gradually changes so that at the
;and of a transition only COLB is shown. This process is called a transition.
;A transition consists of 32 or 64 steps. A single multiplex cycle, whereby
;alternately COLA and COLB are shown, is called a cycle. A cycle has to be
;shorter than 20ms in order for the eye to percieve the two colors as one
;mixed color. The ratio between COLA and COLB is determined by variable
;A2B_RAT in such a way that when A2B_RAT=00 only COLB is shown.
;COLA and COLB are generated using pulse-width modulation. For each color
;2 bits have been used, so that one color can be represented by one byte.
;The format for a color is xxRR YYBB. For example 0011 0000 is full intensity
;red, 0001 0000 represents 33% intensity red, xx11 1111 is white. The code
;1111 1111 represents the end of the table. One pulse width modulation cycle
;is called a burst. With 2 bits, a burst consists of 4 levels. Each cycle
;consists of either 32 or 62 bursts.
;With respect to the original routine a small part has been added: the routine first
;checks if there is a connection to the mains. If not the LEDs are switched of
;immediately to same power in the buffer capacitor. This is needed because the
;way how the time-set button has been implemented
;
rnd_col:        nop             ;
rnd_col_0:      movf    NBURST,W       ;First see if in this burst we have
                xorwf   A2B_RAT,W      ;to show COLA or COLB. The switch
                btfsc   STATUS,Z       ;from COLA to COLB in a cycle is made
                bsf     BITS1,AB       ;when A2B_RAT=NBURST
                movf    COLB,W         ;COLOR becomes the actual color
                movwf   COLOR          ;
                movf    COLA,W         ;
                btfsc   BITS1,AB       ;
                movwf   COLOR          ;

                movf    LCNT,W         ;This part generates the levels in a burst
                xorlw   03FH           ;by switching the LEDs on (and off).
                btfss   STATUS,Z       ;LEDS of at beginning of a new burst
                goto    rnd_col_1      ;
                bcf     PORTA,0        ;turn the LEDs of
                bcf     PORTA,1        ;
                bcf     PORTA,2
rnd_col_1:      movf    COLOR,W        ;xor the color with the level
                xorwf   LCNT,W         ;counter and save in LTMP
                movwf   LTMP           ;
                andlw   03H            ;single out color 1
                btfsc   STATUS,Z       ;skip if color.neq.counter
                bsf     PORTA,0        ;set color 1
                movf    LTMP,W         ;single out color 2
                andlw   0CH            ;
                btfsc   STATUS,Z       ;skip if color.neq.counter
                bsf     PORTA,1        ;set color 2
                movf    LTMP,W         ;single out color 3
                andlw   030H           ;
                btfsc   STATUS,Z       ;skip if color.neq.counter
                bsf     PORTA,2        ;set color 3

                movlw   015H           ;decrement the level counter with 15H
                subwf   LCNT,W         ;LCNT-W -> W
                movwf   LCNT           ;
                btfss   STATUS,Z       ;if zero skip
                return                 ;if not zero return
```

```
                movlw   03FH            ;reset level counter
                movwf   LCNT            ;
                
                incf    NBURST,F        ;in this part we count the number of
                movf    NBURST,W        ;bursts in one cycle. Note that by
                xorlw   NBURST_MAX      ;doing it this way, NBURST=1FH will
                btfss   STATUS,Z        ;immediately result in a reset of
                return                  ;of NBURST, so that COLA will never
                clrf    NBURST          ;be displayed.
                bcf     BITS1,AB        ;
                
                decfsz  NCYCLE,F        ;in this small part, the number of
                return                  ;cycles that make up one transition
                movlw   NCYCLE_REL      ;step are counted. By modifying
                movwf   NCYCLE          ;NCYCLE_REL the transition speed may be modified
                
                incf    A2B_RAT,F       ;after each transition step A2B_RAT
                movf    A2B_RAT,W       ;is incremented. This for every
                xorlw   1+NBURST_MAX    ;transistion step modifies the
                btfss   STATUS,Z        ;COLA/COLB ratio. When we have gone
                return                  ;through all the possible transition
                clrf    A2B_RAT         ;step the next color is loaded.
                
                movf    COLB,W          ;COLB now becomes the starting color
                movwf   COLA            ;COLA, and a new color from the table
                movf    INDX,W          ;replaces COLB. variable INDX points
                call    c_table         ;into the table. When the "color"
                movwf   COLB            ;FFH is found the pointer "loops
                incf    INDX,F          ;around" to the beginning of the
                movf    INDX,W          ;table.
                call    c_table         ;
                xorlw   0FFH            ;
                btfss   STATUS,Z        ;
                return                  ;
                clrf    INDX            ;
                return                  ;return
;
;
;
;
                org     0300H           ;
;include        GROUP1.INC              ;only primary colors
include         GROUP2.INC              ;include also two primary colors
;include        GROUP3.INC              ;include also three primary colors
;
                sEND
```