

Sump_Ver_2-1.asm

```
*****
* Project: Sump Tank Controller *
* *
* Date: Started 10/10/2003 *
* *
* Author: Gordon Wall *
* *
*****
* Description: *
* *
* Simple controller for an upgrade of my variation on the Dunk Tank coined *
* the "Sump Tank". The name is due to the fact that a Sump Pump is used to *
* douse the "victim" rather than dunking them. *
* This implementaton allows adjustment of the pump ON time from 1 to 9 sec, *
* displaying the value on a seven segment display. One push button input *
* allows for changing the time (Default = 5 sec). The target switch is *
* an input, and when this switch closes, the sump pump is turned on for *
* the selected time. *
*****
* Uses a PIC12F629 as a controller *
*****
CONFIG_CP_OFF & _CPD_OFF & _BODEN_OFF & _MCLRE_OFF & _WDT_OFF & _PWRTE_ON &
INTRC_OSC_NOCLKOUT
```

```
#include <p12f629.inc>
; GENERAL PURPOSE REGISTERS
DELAYCNT1 EQU 0x20
DELAYCNT2 EQU 0x21
DELAY_MULT EQU 0x22
Offset EQU 0x23
TX_BUF EQU 0x24
W_TEMP EQU 0x25 ; For saving W Register during Interrupt
CURRENT EQU 0x26 ; Maintain Current Value of Relay Timer
SHADOW EQU 0x27 ; Use Shadow byte for controlling GPIO
TIME_VAL EQU 0x27 ; Copy of Current Time decrements in
Interrupt
HALF_COUNT EQU 0x28 ; Half Second Counter
STATUS_TEMP EQU 0x29 ; Used to save STATUS in ISR
```

```
*****
* Port Assignments I/O Definition *
*****
```

```
#define SHIFT_BIT SHADOW, 0
#define CLK_BIT SHADOW, 1
#define CLR_SR SHADOW, 2 ; Could free up this GPIO
#define SET_TIME GPIO, 3 ; GP3 must be input!!!
#define RELAY_OUT SHADOW, 4 ;
#define TARGET GPIO, 5
#define SW_DELAY 0xFF ; Delay for switch debounce (771 usec)
#define CLK_DELAY 0xFF ; Delay for clk signal (390 usec) 80->FF
#define MULT_DELAY 0x4E ; Mult for sw debounce (total = 60 msec)
```

```
*****
* Reset Vector and Interrupt Vector *
*****
org 0X000
Reset_Vec goto START
org 0x004
Int_Vec
```

Sump_Ver_2- 1. asm

```

bcf      INTCON, GIE          ; Turn off interrupts, Global
bcf      INTCON, PEIE        ; Also Peripheral
movwf   W_TEMP               ; Save W Register
movf    STATUS, w
movwf   STATUS_TEMP          ; Save Status Register
bcf      PIR1, T1IF          ; Clear Timer1 Interrupt Flag
incf    HALF_COUNT, f        ; Each Interrupt is 0.5 sec
btfss   HALF_COUNT, 1        ; If 2 intervals -> decrement TIME_VAL
goto    Press0n              ; Only 0.5 sec
decf    TIME_VAL, f          ; Reduce time by 1 sec
movl w  0x00
movwf   HALF_COUNT

Press0n
movl w  0x0B                  ; Reload Timer1
movwf   TMR1H                 ; Fosc/4 = 1 MHz and a prescale of 8 is used
movl w  0xDB                  ; A value of 0xDB hex gives us an interrupt
movwf   TMR1L                 ; time of 0.5 sec
movf    STATUS_TEMP, w
movwf   STATUS
movf    W_TEMP, w             ; Restore W Register
bsf     INTCON, PEIE
bsf     INTCON, GIE          ; Turn Interrupts back on

retfie

; *****
; * Initialization I/O, Timers, Display *
; *****
START
    clrf   INTCON              ; Disable ALL Interrupts
    call  0x3FF                ; Get Factory Calibration Value
    bsf   STATUS, RP0          ; Bank 1
    movwf OSCCAL              ; Load the Oscillator Value
    bcf   STATUS, RP0          ; Initialize to Bank 0
    clrf  GPIO                 ; Init GPIO
; PIC12F629
    clrf  ADCON0              ; Turn off A/D converter Remove for
    bsf   STATUS, RP0          ; Bank 1
;
    clrf  ANSEL               ; No A/D use make I/O Remove for PIC12F629
    movl w b' 00101000'        ; GP0 Output, GP1 Output, GP2 Output, FIX!
;                                     ; GP3 Input, GP4 Output, GP5

Input
    movwf TRISIO              ; Set direction registers
    bsf   STATUS, RP0          ; Bank 1
    movl w b' 00000000'        ; GPIO pullup set by port latch, rest don't
care
    movwf OPTION_REG
; No weak pullup on GP3 must use hard pullup
    bsf   WPU, 5              ; GP5 - Weak Pullup Enabled
    clrf  VRCON               ; Turn off VREF
    bcf   STATUS, RP0          ; Back to Bank 0
    movl w b' 00000111'
    movwf CMCON               ; Turn off Comparator
; Set up Timer 1, but do not enable the Interrupts until Target is hit
    bsf   STATUS, RP0          ; Bank 1
    movl w b' 00000001'        ; Enable Timer 1 Peripheral Interrupt
    movwf PIE1
;
    bcf   STATUS, RP0          ; Bank 0
    movl w b' 00110101'        ; Setup Timer 1 and enable (8:1 prescale)
    movwf T1CON
    bcf   PIR1, T1IF          ; Clear Timer1 Interrupt Flag
    movl w 0x04                ; Pointer into Table for Hex values
    movwf Offset              ; Note offset of 4 is a time of 5
    movl w 0x05                ; Default time is 5 seconds

```

Sump_Ver_2- 1. asm

```

movwf CURRENT
movl w 0x6D ; Default timer vaue in Hex (5)
movwf TX_BUF ; Moving this to the TRANS_TEST2 Routine

Fails
movl w b' 00101100' ; Initialize GPIO inputs and CLR High!
movwf GPIO
movwf SHADOW ; Also initialize SHADOW Reg
goto TRANSMIT ; Go Display the "5" on 7-Seg Display

```

```

;*****
;*
;* Main Program - watch 2 switches *
;*****

```

```

Main_Loop
btfss SET_TIME ; Check Time Bump Input
goto UpTime ; Change it
btfss TARGET ; Check if Target Hit
goto Pump_On ; Target Hit -> Turn on Pump
goto Main_Loop ; Loop again

```

```

;*****
;*
;* Subroutines Follow *
;*****

```

```

;*****
;*
;* Output Routine *
;*****

```

```

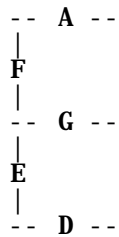
;*****
;*
;* Outputs to 8 bit shift register to update the 7 segment LED display. This *
;* uses only 3 of the pins on the PIC12F629, 1 clock signal, 1 data signal *
;* and 1 clear signal to reset the shift register to zero. Even this pin *
;* could free up as shifting 8 bits into the shift register will clear the *
;* Previous value. This routine is very simple variation of a I2C routine. *
;*****

```

```

;*****
;*
;* Hardware Description
;* 7_Segment Display (Common Cathode)
;*****

```



```

;*
;* Respective shift register outputs are connected to the respective
;* Segments as QA -> A, QB -> B etc. The table below shows the relationship
;* of displayed numbers vs shift register outputs. Also given is the hex
;* representation used by this program.
;*****

```

Shift Register Outputs									
Number	QH	QG	QF	QE	QD	QC	QB	QA	HEX NUMBER
1	0	0	0	0	0	0	1	1	06
2	0	1	0	1	1	0	1	1	5B
3	0	1	0	0	1	1	1	1	4F
4	0	1	1	0	0	1	1	0	66
5	0	1	1	0	1	1	0	1	6D
6	0	1	1	1	1	1	0	1	7D
7	0	0	0	0	0	1	1	1	07
8	0	1	1	1	1	1	1	1	7F

```

; *          9          0 1 1 0 0 1 1 1          67
; *
; *          Sump_Ver_2-1.asm
; *
; *          *****
; *          Time Set Input Handler
; *          Here increment the time and call Transmit to send
; *          the current setting to the 7-segment display
; *          *****
UpTime
    call    SW_TIME          ; Let switch stop bouncing
    call    SW_TIME
    call    SW_TIME
    call    SW_TIME
    incf   CURRENT, f
    incf   Offset, f
    goto   CHK_ROLL

Lookup
    movf   Offset, w
    call   SEG_TABLE
    movwf  TX_BUF          ; Put data in Transmit Buffer
    goto   TRANSMIT

; *          *****
; *          Lookup Table
; *          *****
SEG_TABLE:
    addwf  PCL, f
    retlw  0x06          ; Segment "1"
    retlw  0x5B          ; Segment "2"
    retlw  0x4F          ; Segment "3"
    retlw  0x66          ; Segment "4"
    retlw  0x6D          ; Segment "5"
    retlw  0x7D          ; Segment "6"
    retlw  0x07          ; Segment "7"
    retlw  0x7F          ; Segment "8"
    retlw  0x67          ; Segment "9"

CHK_ROLL
    btfsc  Offset, 3
    goto   Check_Bit1
    goto   Lookup

Check_Bit1
    btfss  Offset, 0
    goto   Lookup

Roll
; *          ; We have a 9 so Roll back to a 0
    movlw  0x00
    movwf  Offset          ; Pointer into table
    movlw  0x01
    movwf  CURRENT        ; Change time to 1 second
    goto   Lookup

; *          *****
; *          Transmit Routines for Display Update
; *          *****
; *          INPUTS:          Current time setting in hex for display
; *          OUTPUTS:        Clocked bit output to shift register
; *          *****
; *          SPECIAL NOTE!
; *          This routine illustrates the correct way to deal with
; *          setting bits on the ports by using a Shadow register
; *          This is the "infamous" Read, Modify, Write issue
; *          that plagues many programmers
; *          *****

```

Sump_Ver_2-1.asm

```

; *****
;
; *****
;   Transmit Loop
; *****

TRANSMIT
    bsf     STATUS, C           ; Set the end of loop flag
    rlf     TX_BUF, f          ; Place first bit into Carry

TX_LOOP
    bcf     SHIFT_BIT          ; Precondition the output
    btfs   STATUS, C           ; Check to see if bit is 0 or 1
    bsf     SHIFT_BIT
    movf   SHADOW, w
    movwf  GPIO                ; Put the data bit out to the Shift Register
    call   CLK_TIME            ; Give the data time to settle
    call   CLK_TIME
    call   BIT_OUT             ; Send the Bit using clock
    bcf     STATUS, C           ; Clear data in Carry
    rlf     TX_BUF, f          ; Put next bit into Carry
    movf   TX_BUF, f           ; Force Zero Bit
    btfs   STATUS, Z           ; Exit?
    goto   TX_LOOP
    goto   Main_Loop

; *****
;   Shift Bits Out to Shift Register
; *****
BIT_OUT
    bsf     CLK_BIT            ; Pos clock edge clocks data
    movf   SHADOW, w
    movwf  GPIO                ; Positive clk edge
    call   CLK_TIME
    bcf     CLK_BIT
    movf   SHADOW, w
    movwf  GPIO                ; Return clock to low
    call   CLK_TIME            ; Allow for Prop Delay @ S/R
    return

Pump_On
    bsf     RELAY_OUT
    movf   SHADOW, w
    movwf  GPIO                ; Positive Relay Signal
    movf   CURRENT, w
    movwf  TIME_VAL            ; Copy of current Time, decrements in Interrupt
    movl   0x00
    movwf  HALF_COUNT         ; Initialize half sec counter
    movl   0x0B
    movwf  TMR1H               ; Load Timer 1
    movl   0xDB
    movwf  TMR1L
    bsf     INTCON, PEIE
    bsf     INTCON, GIE        ; Turn Interrupts on

Watch_Time
    nop
    nop
    nop
    nop
    movf   TIME_VAL, f         ; Check for Zero
    btfs   STATUS, Z           ; Exit?
    goto   Watch_Time         ; No, Keep timing
    bcf     INTCON, PEIE
    bcf     INTCON, GIE        ; Turn Interrupts on
    bcf     RELAY_OUT

```

Sump_Ver_2- 1. asm

```

    bsf      CLR_SR
    movf    SHADOW, w
    movwf   GPIO          ; Release Relay
    goto    Main_Loop
; *****
; Delay Routine for reset and clocking to shift reg *
; *****
CLK_TIME
    movl w   CLK_DELAY          ; Delay of 390 usec
    movwf   DELAYCNT1
Wai t1
    decfsz  DELAYCNT1, f
    goto    Wai t1
    nop
    return
; *****
; Delay Routine for switch debounce (60 msec) *
; *****
SW_TIME
    movl w   SW_DELAY
    movwf   DELAYCNT2          ; Low Count
    movl w   MULT_DELAY
    movwf   DELAY_MULT        ; Mult of Low Count
Wai t2
    decfsz  DELAYCNT2, f
    goto    Wai t2
    decfsz  DELAY_MULT, f
    goto    Wai t2
    nop
    return
SW2_TIME
    movl w   SW_DELAY
    movwf   DELAYCNT2          ; Low Count
Wai t3
    decfsz  DELAYCNT2, f
    goto    Wai t3
    movl w   SW_DELAY
    movwf   DELAYCNT2        ; Low Count
Wai t4
    decfsz  DELAYCNT2, f
    goto    Wai t4
    return
END

```