**PIC12C508 replaces potentiometer and multi-stage switch and increases user-friendliness.**

**APPLICATION OPERATION**

The usual wiper control in a car has two problems, both of which this application can solve. First, it uses too many parts, usually an on/off switch and either a potentiometer to adjust the wiping interval or a multi-stage switch. Second, it is not very user-friendly: You either have a limited number of interval periods or, if the wiper is controlled via a potentiometer, you have to adjust the interval period, watch the windscreen if the interval is sufficient (takes at least one or two times wiping), re-adjust the period and so on.

This application uses a single switch and a PIC12C508 to adjust the wiper interval settings. The main point is that the driver decides when the windscreen is too 'wet'. It is easiest to understand the operation using the attached graphics.

Upon switching the unit on, the windscreen is wiped periodically with a default interval. By switching the unit off, the driver inhibits wiping causing the windscreen to get wetter and wetter. As soon as the driver decides – the windscreen should not get wetter than this– he/she switches it back on. Doing so, the driver sets the new interval according to the time passed between the last clearing of the windscreen and switching the unit back on. This way the driver can either lengthen or shorten the interval to exactly what he wishes - the wipers will not go too fast (by the way, often it is the case in traffic jams, there's just no suitable setting!) and it won't go too slow.

**OPERATION FLOWCHART**

The software shows how to:

- Implement an accurate timer with a period longer than possible with the internal timer (very efficient code! a subroutine which just has to be called every now and then).
- Generate software interrupts.
- Debounce switches in an interrupt routine and protect the software against noise on the switch input.
- Return boolean values (again very efficiently!).

---

Microchip Technology Incorporated, has been granted a nonexclusive, 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 The author has affirmed that this work is an original, unpublished work and that he/she owns all rights to such work.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.
**FLOW CHART**

```
Start

wait till switch closed

wiper interval = default
LED on

wiper interval = time passed since last wipe
acknowledge interval set with short beep

wiper interval < minimum ?

yes

wiper interval = default

delay 30 seconds

interrupt if switch is closed again

no

wipe continuously
interrupt if switch is opened

stop wiping

LED off

initiate single wipe

delay wiper interval

interrupt if switch is opened

Microchip Tools Used
Assembler/Compiler Version:
MPASM V1.4

**Graphical Hardware Representation**

```
Vcc    1    PIC12C508
LED    2    Out
Buzzer* 3    Out
Switch 4    Out
```

*Open drain in order to make it shareable with other units.
APPENDIX A:  SOURCE CODE

rocessor 12c508
radix dec
include "p12c508.inc"
#define __12C508
__config _WDT_OFF & _IntRC_OSC & _MCLRE_OFF & _CP_ON
#define zero STATUS, 2
#define carry STATUS, 0
#define TRUE  0
#define FALSE   -1

CBLOCK 0x07 ; start of RAM
ENDC

MOVWF OSCCAL
GOTO Main

TMR0overrun EQU 16384 ; timer0 overrun every 16.4ms
#define   ms    1000/TMR0overrun ; in the main program when changing
#define secs 1000000/TMR0overrun

;* Hardware ***************************************************************

#define Switch GPIO, 3
#define LED GPIO, 5
#define Buzzer GPIO, 4
#define LedOn GPIO, 5
#define LedOff GPIO, 4

BuzzerOn MACRO
BCF Buzzer ; Buzzer is open drain in order
MOVLW b'001000' ; to share it with other units
TRIS GPIO
ENDM

BuzzerOff MACRO
MOVLW b'011000'
TRIS GPIO
ENDM

OutputOn MACRO
MOVLW b'111000' ; output is activ low
ANDWF GPIO
ENDM

OutputOff MACRO
MOVLW b'000111'
IORWF GPIO
ENDM

WiperThreshold EQU 500*ms
MinimumInterval EQU 1*secs
DefaultInterval EQU 2*secs
BeepLength EQU 200*ms
DenoiseTime EQU 50*ms
DebounceTime EQU 50*ms
DisableAfter EQU 20*secs

;/* Macros *******************************************************/

TWSTrue MACRO ; (T)est (W) and (S)kip if (True)
    IORLW 0
    BTFSS zero
ENDM

TWSFalse MACRO
    IORLW 0
    BTFSC zero
ENDM

#define SkipIfZero BTFSS zero
#define DoIfZero BTFSC zero
#define RET RETLW 0

;/* Switch ***********************************************************/

CBLOCK
Denoise
Debounce
Flags
ENDC

#define SwitchClosed Flags, 0

.HandleSwitch MACRO
    BTFSS Switch
    GOTO HS.closed

    MOVWF Debounce
    ; switch open now, so
    ; reset timer for 'switch closed'
    INCFSZ Timer1L
    GOTO Timer1.done

    BTFS SwitchClosed
    GOTO HS.done
    ; switch is already denoised
    DECFSZ Denoise
    ; otherwise, wait till switch
    BCF SwitchClosed
    GOTO HS.done

    ; is stable a certain time

    BSF SwitchClosed
    GOTO HS.done

HS.opened MOVLW DebounceTime
            ; as above
    MOVWF Denoise
    BTFS SwitchClosed
    GOTO HS.done

    DECFSZ Debounce
    GOTO HS.done

    BSF SwitchClosed
    GOTO HS.done

HS.closed MOVLW DenoiseTime
           ; as above
    MOVWF Denoise
    BTFS SwitchClosed
    GOTO HS.done

    DECFSZ Debounce
    GOTO HS.done

    BSF SwitchClosed
    GOTO HS.done

HS.done ENDM

;/* Timer ***********************************************************/

CBLOCK
Timer0L
Timer0H
Timer1L
Timer1H
OldTMRO
ENDC

.IncreaseTimer1 MACRO
    INCFSZ Timer1L
    GOTO Timer1.done
INCF Timer1H
ENDM

Interrupt    MOVF OldTMR0, W   ; increase Timer on TMR0-overflow
           SUBWF TMR0, W   ; overflow, if OldTMR0 > TMR0
           BTFSC carry
           GOTO Interrupt.done
           ADDWF OldTMR0
           ; program enters here every 16.4ms
           HandleSwitch
           IncreaseTimer1
           INCFSZ Timer0L
           RETLW FALSE
           INCFSZ Timer0H
           RETLW FALSE
           RETLW TRUE   ; return TRUE upon hitting zero
           ; in timer0 !

Interrupt.done   ADDWF OldTMR0
           RETLW FALSE

LoadTimer0 MACRO Value
           MOVLW low(-Value)
           MOVWF Timer0L
           MOVLW high(-Value)
           MOVWF Timer0H
           ENDM

;* Subroutines ************************************************************

Beep           LoadTimer0 BeepLength
Beep.loop     BTFSC TMR0, 2   ; this will generate about 2 kHz
           BuzzerOn
           BTFSS TMR0, 2
           BuzzerOff
           CALL Interrupt
           TWSTrue
           GOTO Beep.loop
           BuzzerOff
           RET

Delay         MACRO Value
           LOCAL Loop
           LoadTimer0 Value
           Loop
           CALL Interrupt
           TWSTrue
           GOTO Loop
           ENDM

;******************************************************************************

CBLOCK
IntervalL
IntervalH
ENDC

Main         MOVLW b'10010101' ; pullups on
           OPTION      ; -> TMR0overflow every 16.384us
           BuzzerOff   ; this will also set TRIS correctly
Electromechanical Switch Replacement

OutputOff

MOVLW DebounceTime
MOVWF Debounce
BCF SwitchClosed

;

Main.loop  LedOff

CALL Interrupt
BTFSS SwitchClosed
GOTO Main.loop

MOVLW low(DefaultInterval); Interval:= DefaultInterval
MOVWF IntervalL ;
MOVLW high(DefaultInterval); 
MOVWF IntervalH :
LedOn

CheckInterval  MOVWLW high(MinimumInterval); if Interval<MinimumInterval
SUBWF IntervalH, W ; then GOTO Wipe.continuous
BTFSS carry ;
GOTO Wipe.continuous;
BTFSS zero ;
GOTO Wipe.interval ;
MOVLW low(MinimumInterval); 
SUBWF IntervalL, W ;
BTFSS carry ;
GOTO Wipe.continuous;

;-------------------------------;
Wipe.interval  OutputOn ; initiate single wipe
CLRF Timer1L ; always clear timer1 upon
CLRF Timer1H ; wiping
Delay WiperThreshold;
OutputOff

Wipe.int.loop  CALL Interrupt

BTFSS SwitchClosed
GOTO Off?

MOVF IntervalH, W ; if timer1<Interval
SUBWF Timer1H, W ; then GOTO Wipe.int.loop
BTFSS carry ; else GOTO Wipe.interval
GOTO Wipe.int.loop ;
BTFSS zero ;
GOTO Wipe.interval ;
MOVF IntervalL, W ;
SUBWF Timer1L, W ;
BTFSS carry ;
GOTO Wipe.int.loop ;
GOTO Wipe.interval ;

;-------------------------------;
Wipe.continuous OutputOn

CLRF Timer1L ; always clear timer1 upon
CLRF Timer1H ; wiping
CALL Interrupt
BTFSC SwitchClosed
GOTO Wipe.continuous
OutputOff

;-------------------------------;
Off?
Off?.loop  BTFSC SwitchClosed
GOTO NewIntervalSet
CALL Interrupt
TWSTrue
GOTO Off?.loop
GOTO Main.loop ; 30 seconds expired

;-----------------------------;

NewIntervalSet  MOVF Timer1L, W
MOVWF IntervalL
MOVF Timer1H, W
MOVWF IntervalH

CALL Beep
GOTO CheckInterval

;**************************************************************************

END