INTELLIGENT PUSH BUTTON FOR AIR CONTROL AND MORE

When driving behind busses, big trucks or lorries, the ventilation system fills your car with stuffy exhaust fumes. Therefore, many cars are equipped with a mode to circulate the interior air. Usually, one forgets to disable this mode afterwards and the air is no longer replaced with fresh air. This smart switch solves the problem mentioned while maintaining full functionality of the existing system - utilizing a single push button.

(There is the same problem with the rear window defrosting system: The window usually is clean after a few minutes, but one always forgets to deactivate the defroster.)

APPLICATION OPERATION

Since a user usually needs the device (ventilation, defroster, etc.) only for a certain period of time, this application features a timer function.

STANDARD OPERATION

Press the push button to activate the device (a short beep and/or an LED going on will acknowledge this). After a defined period of time, the device is automatically deactivated (signaled by a short beep and five seconds of a blinking LED – showing the driver which device is currently off – in case there are both ventilation and defroster control or even more).

Note: So the 'interior air circulation' mode cannot be forgotten and will not stay active longer than necessary.

Alternatively, the driver can activate the device permanently by pressing the button for about one second (the PICmicro™ MCU will acknowledge this with a long beep and an LED going on). To switch the device off again, just hit the push button (signaled by a beep and an LED going off).

Note: The timer period may be interrupted by just pressing the button.

The actual usage is much more intuitive than this explanation might suggest (all one has to know is: short press - standard function, longer press - permanently on)! See flowchart.

The software on the following pages show how to:

- Debounce buttons
- Implement an accurate timer with a period longer than is possible with the internal timer (very efficient code – a subroutine which just has to be called every now and then)
- Return boolean values (again very efficiently!)
FLOW CHART

GRAPHICAL HARDWARE REPRESENTATION

*Shared with other units
APPENDIX A: SOURCE CODE

;******************************************************************************
;* Project: SmartSwitch                                              *
;******************************************************************************

processor 12c508
radix dec
#include "p12c508.inc"
#define __12C508
#define _config _WDT_ON & _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

GOTO Main

;* Hardware *****************************************************************
#define Button GPIO, 3
#define LED GPIO, 5
#define Buzzer GPIO, 4
LedOn MACRO
BCF LED ; LED output is activ low
ENDM
LedOff MACRO
BSF LED
ENDM
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

#define LongTime 800 ; Button down longer than this (in ms)
#define Period 5*6000 ; otherwise output is this long on

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

© 1997 Microchip Technology Inc.
;* Timer *********************************************************************
; Usage: LoadTimer xx ; Load timer with value xx ms
;   HandleTimer   ; execute this MACRO at least every
; TMR0overrun us ; and it will adjust the timer-variable
; correctly ; and return true upon hitting zero
#define TMR0overrun 16384 ; timer0 overrun every 16.4ms
; remember to change the option value
; in the main program when changing
; this
 CBLOCK
 TimerL
 TimerH
 OldTMR0
 ENDC

HandleTimer_ MOVF OldTMR0, W ; increase Timer on TMR0-overflow
 SUBWF TMR0, W ; overflow, if OldTMR0 > TMR0
 BTFSC carry
 GOTO HT_done
 INCFSZ TimerL
 GOTO HT_done
 INCFSZ TimerH
 GOTO HT_done
 ADDWF OldTMR0
 RETLW TRUE ; return TRUE upon hitting zero
 HT_done
 ADDWF OldTMR0
 RETLW FALSE
;
LoadTimer MACRO Value
 MOVLW low(-1000*Value/TMR0overrun)
 MOVWF TimerL
 MOVLW high(-1000*Value/TMR0overrun)
 MOVWF TimerH
 ENDM

HandleTimer MACRO ; The one routine is a MACRO, the
 can
 CALL HandleTimer_; other a subroutine - who
 ENDM; remember ? So make both a MACRO !
;
Wait500ms MACRO ; Due to the stack depth this is
 LOCAL Wait.loop; needed as a MACRO
 LoadTimer 500
 Wait.loop HandleTimer
TWSTrue
GOTO Wait.loop
ENDM

* Subroutines  ******************************************

CBLOCK
Counter ; for various counter-loops
ENDC

ShortBeep   LoadTimer 200 ; 200ms beep
GOTO Beep

LongBeep   LoadTimer 800 ; 800ms beep

Beep
BTFSC TMR0, 2 ; this will generate about 2 kHz
BuzzerOn
BTFSS TMR0, 2
BuzzerOff
HandleTimer
TWSTrue
GOTO Beep
BuzzerOff
RET

LedBlink   MOVLW 5
MOVWF Counter

LedBlink.loop   LedOn
Wait500ms
LedOff
Wait500ms
DECFSZ Counter
GOTO LedBlink.loop
RET

WaitTillPressed  LoadTimer 50; reload timer with 50 ms ...
WTP.loop   BTFSC Button
GOTO WaitTillPressed; ... while button not pressed
HandleTimer
TWSTrue
GOTO WTP.loop
RET

WaitTillReleased LoadTimer 50; reload timer with 50 ms ...
WTR.loop   BTFSS Button
GOTO WaitTillReleased; ... while button pressed
HandleTimer
TWSTrue
GOTO WTR.loop
RET

Main   MOVWF OSCCAL
MOVLW b'10010101'; pullups on
OPTION ; -> TMR0overrun every 16.384us
BuzzerOff ; this will also set TRIS correctly
OutputOff
LedOff
Main.loop    CALL WaitTillPressed
            OutputOn
            LedOn
            LoadTimer LongTime; let's see whether button is held that long ...
            GOTO ModeShort
            HandleTimer
            TWSTrue
            GOTO CheckMode

CheckMode    BTFSC Button
            GOTO ModeShort
            HandleTimer
            TWSTrue
            GOTO CheckMode

ModeShort    CALL ShortBeep; Mode: period on
            LoadTimer Period
            ModeShort.loop  BTFSS Button
            GOTO ManualOff
            HandleTimer
            TWSTrue
            GOTO ModeShort.loop
            OutputOff
            CALL ShortBeep
            CALL LedBlink
            GOTO Main.loop

ModeLong     CALL LongBeep; Mode: permanently on
            CALL WaitTillReleased; Button maybe still pressed
            CALL WaitTillPressed
            GOTO ManualOff

ManualOff    OutputOff
            LedOff
            CALL LongBeep
            CALL WaitTillReleased
            GOTO Main.loop

;*****************************************************************************
END