Automate the Home

Home Security System

Author: *Jim Nagy*
Ontario, Canada
email: nagy@wwdc.com

APPLICATION OPERATION:

The small size, and low cost of PIC12C508 controllers make them very suitable for use as the heart of home/ small business security systems. This design demonstrates one such system that can easily be made. Additional functionality (monitored loops) and/or redundancy can be obtained if desired, by duplicating this circuit and possibly combining the outputs.

Inputs from a normally closed 'delayed' loop, from a normally closed 'instantaneous' loop, and from a normally open 'panic button' circuit are all monitored by the PIC12C508, and used to drive two alarm outputs.

The delayed loop is configured to provide a 45 second delay on alarm in order to allow the user sufficient time to enter or exit the monitored zone. Typically, this loop would be used for magnetic door switches or for infrared body sensors. During this 45 second period, the alarm LED pulses on and off rapidly to warn of impending activation.

The instantaneous loop provides no appreciable delay (other than a 0.5 sec debounce period) before causing the alarm to sound, unless the system has just been armed and 45 seconds has not yet elapsed. During this period, the loop acts like the delayed loop allowing the user time to check the loop monitoring LEDs and exit. This loop would typically be used to monitor other points such as windows.

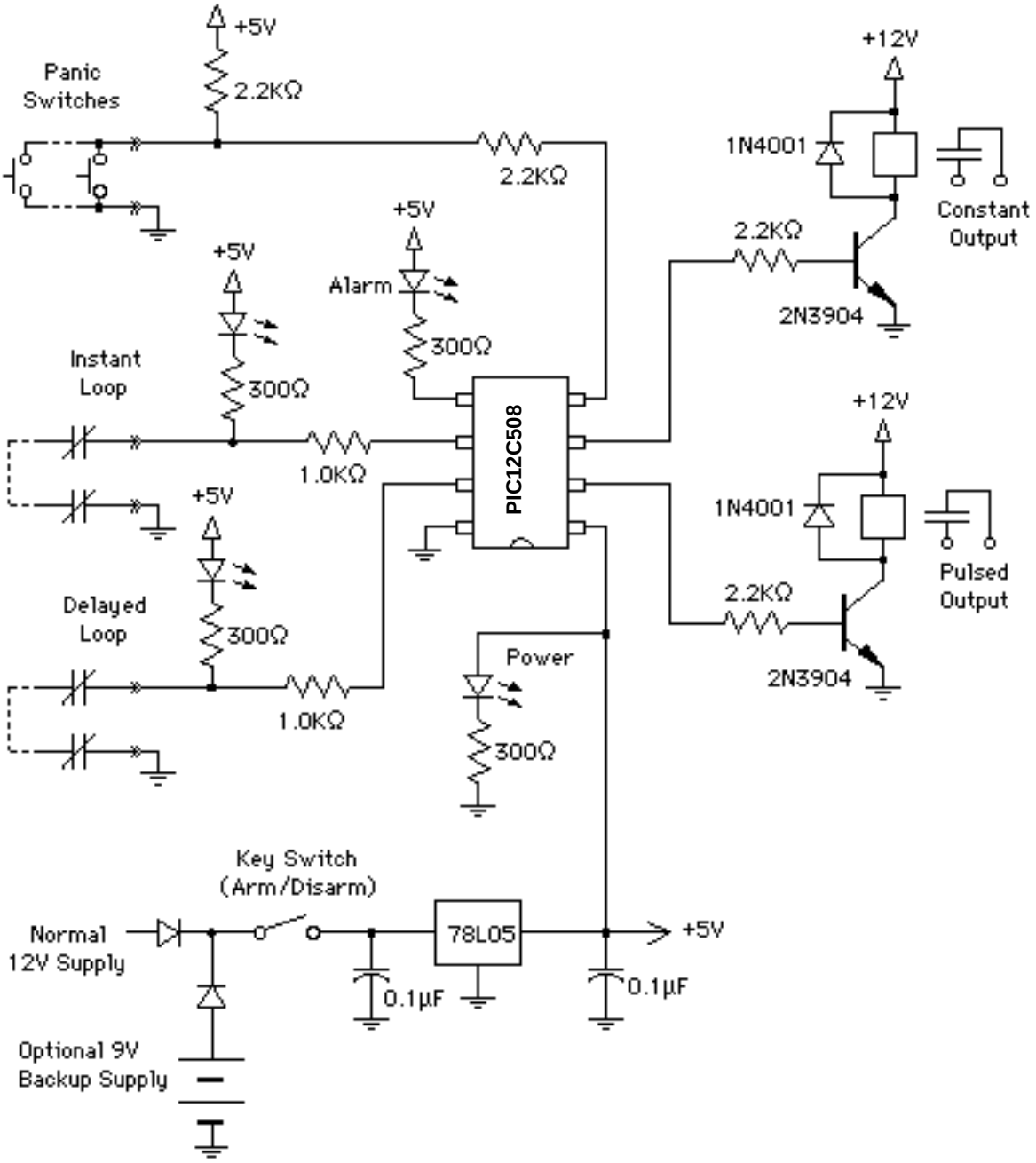
The final type of trigger is from a panic button circuit. As many switches as are required can be wired in parallel, and placed in convenient locations for manually triggering the alarm, at any time.

All three triggers cause the alarm outputs to go to active levels. On alarm, the pulsed output (pin 2) will alternate between +5V, for about1.5 sec, and 0V for 0.5 sec, while the constant output (pin 3) will simply go to +5V. The pulsed output can be used to drive a siren circuit, while the constant output could be useful for turning lights on, etc. Both will remain in this alarm state until reset by the removal of power (typically through a key switch) or for 5 minutes, whichever occurs first.

There is a possibility that an alarm may have occurred and been automatically reset by the system without the knowledge of the user. In order to signal this condition, during the alarm period and afterwards until the system is reset, the LED output (pin 5) will be driven alternately low for 1.5 sec, and high for 0.5 sec, mimicking the pulsed alarm output. This pulsing of the LED output will have no effect on the normal operation of the system, though.

This application uses the WatchDog Timer for automatic reset on software errors, and the internal oscillator for system timing. Power is connected +5V (Vdd) to pin 1 and 0V (Vss) to pin 8. The normally closed delayed loop is connected between Vss and pin 7, while the normally closed instantaneous loop is wired between Vss and pin 6. The panic switch circuit is connected between pin 4 and Vss.

Graphical hardware representation:



APPENDIX A: SOURCE CODE

```
;
;                                       Home Alarm
;                                       ==========
;
;                                                       by Jim Nagy,  November 1997
;
;     A program to use the PIC12C508 as a security alarm system controller. Provides
;     three different monitored inputs (two normally closed, and one normally open) and
;     two outputs (one pulsed, one steady). A LED drive output is provided as well for
;     driving a warning/alarm LED. Pin connections are as follows:
;
;     Pin 1:    Vdd (+5V)
;
;     Pin 2:    Active high pulsed alarm output (+5V for 1.5sec and 0V for 0.5 sec
;               when in alarm). Resets after 5 mins.
;
;     Pin 3:    Active high constant alarm output (+5V). Resets after 5 mins.
;
;     Pin 4:    Normally open input. Typically used for panic switches connected
;               between this pin and Vss. Momentary connection to Vss will cause an
;               alarm to occur. Alarm resets after 5 minutes, if the circuit is open.
;
;     Pin 5:    Active low LED drive output. Pulses rapidly to warn that the system
;               is in the 45sec. delayed alarm period, or pulses slowly if an alarm is
;               in progress/has occurred.
;
;     Pin 6:    Normally closed loop input. Has 0.5 sec debounce period on change of
;               input to reduce the possibility of nuisance triggers. Initiates an alarm
;               immediately after the loop is found to be open, but will not retrigger
;               should the loop remain open. Not enabled during the first 45 seconds
;               after the alarm is first turned on. Typically used for window switches.
;
;     Pin 7:    Similar to pin 6, but the alarm is delayed by 45 seconds, to allow time
;               for entry/exit. Typically used for door switches.
;
;     Pin 8:    Vss (0V)
;
;*******************************************************

; Program Equates
DelayIn      EQU 0              ; GPIO pin names
InstIn       EQU 1
LED          EQU 2
PanicIn      EQU 3
Const        EQU 4
Pulse        EQU 5

DBPanic      EQU 12             ; panic button debounce for 12 *2.048mS
DBDelay      EQU 244            ; delay loop -> 500mS debounce
DBInst       EQU 244            ; instantaneous loop -> 500mS debounce

; bit numbers for flags
TFlag        EQU 7              ; timer will need service
AFlag        EQU 6              ; an alarm has occurred
DFlag        EQU 5              ; in delayed (45 sec) alarm mode
DOpen        EQU 4              ; the delay loop just opened
IOpen        EQU 3              ; the inst loop just opened

; Standard Equates
W            EQU 0
F            EQU 1

GPWUF        EQU 7
```

```
PA0             EQU 5
TO              EQU 4
PD              EQU 3
Z               EQU 2
Zero            EQU 2
DC              EQU 1
C               EQU 0
Carry           EQU 0
; Fuses
MCLRDisabled    EQU 0
MCLREnabled     EQU H'10'
CodeProtect     EQU 0
NoCodeProtect   EQU H'08'
WDTDisabled     EQU 0
WDTEnabled      EQU H'04'
IntRCOsc        EQU H'02'
ExtRCOsc        EQU H'03'
XTOsc           EQU H'01'
LPOsc           EQU 0

; '508 Register Assignments
INDF            EQU H'00'
TMR0            EQU H'01'
PCL             EQU H'02'
STATUS          EQU H'03'
FSR             EQU H'04'
OSCCAL          EQU H'05'
GPIO            EQU H'06'

; program variables
Flags           EQU H'07'      ; storage space for messages
ThisSw          EQU H'08'      ; loop-switch status

; switch0 - the Delayed Loop
LastSw0         EQU H'09'      ; last value - for detecting change
SwState0        EQU H'0A'      ; the official (debounced) status
DBTimer0        EQU H'0B'      ; counter used during debouncing of the switch

; switch1 - the Instant Loop
LastSw1         EQU H'0C'      ;      ""
SwState1        EQU H'0D'      ;      ""
DBTimer1        EQU H'0E'      ;      ""

; switch2 - the Panic Switches
LastSw2         EQU H'0F'      ;      ""
SwState2        EQU H'10'      ;      ""
DBTimer2        EQU H'11'      ;      ""

; The timer bytes
TimerLo         EQU H'12'      ; Lo byte of the timer
TimerMid        EQU H'13'      ; Mid  ""
TimerHi         EQU H'14'      ; High ""

;    The ID words...

           ORG H'0200'
ID0        Data.W   H'0000'
ID1        Data.W   H'0000'
ID2        Data.W   H'0003'
ID3        Data.W   H'000F'

;    and the Fuse bits...

           ORG      H'0FFF'
CONFIG     Data.W    MCLRDisabled + NoCodeProtect + WDTEnabled + IntRCOsc
```

```
;    ******************************************
;          Power on jumps to here...
;    ******************************************

       ORG      H'00'
       MOVWF    OSCCAL       ; store the factory osc. calibration value
       GOTO     Init         ; and jump past the subroutines



;    ******************************************
;       The Subroutines...

;    ******************************************
;       DoTime
;    Updates all of the timers, and controls the LED output
;    Returns with W = 0 if there was no time change, and W = 1 if there was.
;    Must be called at least once every msec to catch Timer0 overflows.

DoTime  BTFSS    TMR0,7             ; high bit of timer set?
       GOTO     dt1                ; no - we've overflowed
       BSF      Flags,TFlag        ; yes, set the flag
       RETLW    0
dt1    BTFSS    Flags,TFlag        ; have the timers been serviced?
       RETLW    0                  ; yes - that's all

;    when Timer0 overflows, all timers are serviced...
       INCF     DBTimer0,F         ; increment the debounce timers
       INCF     DBTimer1,F
       INCF     DBTimer2,F
       INCF     TimerLo,F    ; and the delay timers
       BTFSC    STATUS,Zero
       INCF     TimerMid,F
       BTFSC    STATUS,Zero
       INCF     TimerHi,F
       BCF      Flags,TFlag        ; reset the timer service flag,
       CLRWDT                      ; and the WatchDog timer

       BTFSC    Flags,AFlag        ; has there been an alarm?
       GOTO     SlowFlash    ; if so, flash the LED
       BTFSS    Flags,DFlag        ; are we in delayed entry/exit?
       RETLW    1                  ; no, just return

FastFlash
       BTFSS    TimerLo,5    ; for the fast flash rate
       BCF      GPIO,LED     ; match LED drive to the 64ms bit
       BTFSC    TimerLo,5
       BSF      GPIO,LED
       RETLW    1
SlowFlash
       MOVF     TimerMid,W         ; for slow rate, use 0.5 seccounter
       ANDLW    B'00000011'        ; but only the low two bits of it
       BTFSS    STATUS,Zero        ; to drive the LED
       BCF      GPIO,LED
       BTFSC    STATUS,Zero
       BSF      GPIO,LED
       RETLW    1


;    ******************************************
;       DelayChk
;    Gets the current (debounced) status of the loop connected to GP0 (pin7)
;    Returns with W=1 if there's a change in state, W=0 otherwise, and
;    sets the DOpen bit of Flags set when the loop opens.

DelayChk
```

```
            CLRF     ThisSw             ; assume loop-switch is not open
            BTFSC    GPIO,DelayIn  ; check sw status
            INCF     ThisSw,F      ; the loop is open

;    Compare to last state
            MOVF     LastSw0,W
            XORWF    ThisSw,W
            BTFSS    STATUS,Zero        ; Zero is set if there's been no change
            GOTO     dc1
;    No change since last scan, but are we in a debounce phase?
            MOVF     SwState0,W
            XORWF    ThisSw,W      ; compare present state to the official state
            BTFSC    STATUS,Zero        ; Zero is Clr if they differ(we're in debounce)
            RETLW    0

;    Input is changing, have we gone past the debounce time?
            MOVLW    DBDelay            ; get the debounce time
            SUBWF    DBTimer0,W         ; and compare to elapsed
            BTFSS    STATUS,Carry ; Carry is set if DBTimer0 >= DBDelay
            RETLW    0

;    we've exceeded the debounce time - change the official state of theloop
            MOVF     ThisSw,W
            MOVWF    SwState0      ; store current state, and
            BTFSS    STATUS,Zero        ; check if it's zero (loopnormal)
            BSF      Flags,DOpen        ; if not, set the DOpen bit
            RETLW    1

;    the input is changing state - prepare to debounce
dc1         MOVF     ThisSw,W
            MOVWF    LastSw0            ; remember this passes' state
            CLRF     DBTimer0      ; and reset the debounce timer
            RETLW    0

;    ****************************************
;          InstChk
;    Gets the current (debounced) status of the loop connected to GP1 (pin6)
;    Returns with W=1 if there's a change in state, W=0 otherwise, and
;    sets the IOpen bit of Flags set when the loop opens.

InstChk
            CLRF     ThisSw             ; assume loop-switch is not open
            BTFSC    GPIO,InstIn        ; check sw status
            INCF     ThisSw,F      ; the sw was pressed

;    Compare to last state
            MOVF     LastSw1,W
            XORWF    ThisSw,W
            BTFSS    STATUS,Zero        ; Zero is set if there's been no change
            GOTO     ic1

;    No change since last scan, but are we in a debounce phase?
            MOVF     SwState1,W
            XORWF    ThisSw,W      ; compare present state to the official state
            BTFSC    STATUS,Zero        ; Zero is Clr if they differ (we're in debounce)
            RETLW    0

;    Input is changing, have we gone past the debounce time?
            MOVLW    DBInst             ; get the debounce time
            SUBWF    DBTimer1,W         ; and compare to elapsed
            BTFSS    STATUS,Carry ; Carry is set if DBTimer1 >= DBInst
            RETLW    0

;    we've exceeded the debounce time - change the official state of the loop
            MOVF     ThisSw,W
            MOVWF    SwState1      ; store current state and
```

```
        BTFSS    STATUS,Zero        ; check if it's zero (loopnormal)
        BSF      Flags,IOpen        ; if not, set the IOpen bit
        RETLW    1

;    the input is changing state - prepare to debounce
ic1     MOVF     ThisSw,W
        MOVWF    LastSw1            ; remember this passes' state
        CLRF     DBTimer1     ; and reset the debounce timer
        RETLW    0


;    ******************************************
;       PanicChk
;    Gets the current (debounced) status of the switches connected to GP3(pin 4)
;    Internally stored as SwState2=1 if a panic switch is pressed (input =0V),
;    and SwState2=0 if no switch is pressed
;    Always returns with W = 0, and Zero bit of Carry set for no switch pressed

PanicChk
        CLRF     ThisSw             ; assume switch is not pressed
        BTFSS    GPIO,PanicIn   ; check sw status
        INCF     ThisSw,F     ; the sw was pressed

;    Compare to last state
        MOVF     LastSw2,W
        XORWF    ThisSw,W
        BTFSS    STATUS,Zero        ; Zero is set if there's been no change
        GOTO     pc1

;    No change since last scan, but are we in a debounce phase?
        MOVF     SwState2,W
        XORWF    ThisSw,W     ; compare present state of sw to the official state
        BTFSC    STATUS,Zero        ; Zero is Clr if they differ (we're in debounce)
        GOTO     pc2

;    Switch is changing, have we gone past the debounce time?
        MOVLW    DBPanic            ; get the debounce time
        SUBWF    DBTimer2,W         ; and compare to elapsed
        BTFSS    STATUS,Carry ; Carry is set if DBTimer2 >= DBPanic
        GOTO     pc2

;    we've exceeded the debounce time - change the official state of the switch
        MOVF     ThisSw,W
        MOVWF    SwState2     ; store current state in SwState
        GOTO     pc2

;    the switch input is changing state - prepare to debounce
pc1     MOVF     ThisSw,W
        MOVWF    LastSw2            ; remember this passes' state
        CLRF     DBTimer2     ; and reset the debounce timer
;    we're done for this pass...
pc2     MOVF     SwState2,F         ; check the switch state,
        RETLW    0                  ; and return

;    ******************************************
;    ******************************************
;
;    Here's where it all begins...
;
;    ******************************************
; Set up the Option Register and the IO port...
Init    MOVLW    B'10000010'        ; use intclock input, /8 prescaler
        OPTION                      ; pullups on, and no wakeup on change
        CLRF     GPIO               ; output all 0s on a powerup
        MOVLW    B'00001011'        ; GP0, GP1 and GP3 are inputs,
        TRIS     GPIO         ; GP2, GP4 and GP5 are outputs
```

```
; Clear the variable RAM...
           MOVLW    H'1F'        ; point to the last RAM location
           MOVWF    FSR
cr1        CLRF     INDF         ; and zero it
           DECF     FSR,F
           MOVF     FSR,W              ; loop until FSR points to Register6,
           XORLW    B'11100110'        ; allowing that FSR<7:5> are always 1s
           BTFSS    STATUS,Zero
           GOTO     cr1
; For the first 45 secs, only the panic switch input is active...
           BSF      Flags,DFlag        ; fast flash the LED
wait1      CALL     DoTime
           CALL     PanicChk           ; check for a panic switch
           BTFSS    STATUS,Zero
           GOTO     Alarm
           MOVLW    D'86'        ; check for timeout
           SUBWF    TimerMid,W         ; 86 counts is ~ 45secs
           BTFSS    STATUS,Carry
           GOTO     wait1        ; keep looping until time is up
           BCF      Flags,DFlag        ; stop the flashing LED
           BSF      GPIO,LED     ; and make sure it's off


; Now all three loops are active, watch them forever...
Main       CALL     DoTime       ; keep clock 'running'

           CALL     PanicChk           ; see if a panic switch is pressed
           BTFSS    STATUS,Zero        ; skip on if it isn't
           GOTO     Alarm              ; and alarm if it is

           CALL     InstChk            ; check the Inst loop
           BTFSC    Flags,IOpen        ; and alarm on opening
           GOTO     Alarm

           CALL     DelayChk     ; check the delayed loop
           BTFSS    Flags,DOpen        ; and alarm on opening
           GOTO     Main         ; else, just loop


; Waiting 45 Seconds before alarming...
DelAlarm
           CLRF     TimerLo            ; reset the timer
           CLRF     TimerMid     ; (saves doing arithmetic)
           CLRF     TimerHi
           BSF      Flags,DFlag        ; fast flash the LED
da1        CALL     DoTime
           CALL     PanicChk           ; check for a panic switch
           BTFSS    STATUS,Zero
           GOTO     Alarm
           CALL     InstChk            ; or a break in the Inst loop
           BTFSC    Flags,IOpen        ; and alarm on opening
           GOTO     Alarm
           MOVLW    D'86'        ; check for timeout
           SUBWF    TimerMid,W         ; 86 counts is ~ 45 secs
           BTFSS    STATUS,Carry
           GOTO     da1                ; keep looping until time is up

; It's an alarm, turn the outputs on... (for 5 mins max)
Alarm
           BSF      Flags,AFlag        ; there's been an alarm, flagit
           CLRF     TimerLo            ; reset the timer
           CLRF     TimerMid
           CLRF     TimerHi

           BSF      GPIO,Const         ; turn the continuous output on
```

```
al1     CALL     DoTime
        CALL     PanicChk     ; still have to keep current
        CALL     InstChk           ; on the status of the inputs
        CALL     DelayChk
        MOVF     TimerMid,W
        ANDLW    B'00000011'
        BTFSC    STATUS,Zero       ; set the pulsed output based on the
        BCF      GPIO,Pulse        ; low two bits of the 0.5 sec timer
        BTFSS    STATUS,Zero
        BSF      GPIO,Pulse
        MOVLW    D'02'        ; check for >5 mins
        SUBWF    TimerHi,W    ; ~2*134sec + 60*0.5sec
        BTFSS    STATUS,Carry
        GOTO     al1               ; keep looping until timed out
        MOVLW    D'60'
        SUBWF    TimerMid,W
        BTFSS    STATUS,Carry
        GOTO     al1

        BCF      GPIO,Const        ; >5 mins - stop the alarms
        BCF      GPIO,Pulse
        BCF      Flags,DOpen       ; reset the change of state bits
        BCF      Flags,IOpen
        GOTO     Main         ; and start all over

END
```

NOTES: