Discrete Logic Replacement

Garage Door Indicator

Author: Brian Iehl
Hoffman Estates, Illinois
email: brian@dls.net

INTRODUCTION:

This project displays the status of a garage door. If you have a detached garage like mine, you may not be able to see whether or not the garage door is closed from the house. I finally got tired of walking out of the house and around the garage to see if the door was closed before I turned in for the night, so I designed a circuit to allow me to see if it's shut without me going out into the cold Chicago winter night. This simple battery powered circuit displays either the word OPEN or SHUT. It consists of the LCD display, battery and control circuitry housed in a small case that I place in my kitchen. The other part is a remote switch that closes when the door is shut and opens when the door is open. Having this indicator makes sure I don't give any prowlers an unfair advantage.

APPLICATION OPERATION:

Since this device sits on the window sill in my kitchen, an AC outlet is not available. Therefore, for convenience and simplicity sake, I wanted this project to be battery driven. I also set a goal of having battery life of at least 6 months. A quick check of the Digi-Key catalog showed standard AA size alkaline batteries have a capacity of 170 hours with a load of 100 ohms continuous. For a 1.5 V cell this gives 1.5V / 100 = 15 mA for 170 hours, which is equivalent to 1 mA for 2550 hours. For this project we are using 2 AA batteries in series to give 3 V. This should give us the same capacity, only at the higher voltage. Six months equals 4320 hours (6 * 30 * 24). Therefore the average current drain of this circuit needs to be less than 2550 mA Hr / 4320 Hr = 0.59 mA. This was accomplished with the original discrete logic circuit. The PIC12C508 draws < 2 mA current when running at 4 MHz. The display draws practically nothing. The current drain is basically, a function of the frequency the device is clocked at. By running the oscillator at a lower frequency, we can reduce the current significantly. Here we are running at 200 kHz, therefore we can estimate the current drain to be: 200 kHz * 2 mA / 4 MHz = 0.1 mA. The estimated battery life is then: 2550 mA Hr / 0.1 mA = 25500 hours. This is almost 3 years!

Another goal is for this project to be low cost, since admittedly this project performs a relatively simple task, it does not warrant paying a high price for it. Also, it is always more challenging and fun to design for low cost than simply throwing money at a problem. The big expense with this type of project is usually the display. Especially, if you want half inch high characters so you can read the display from across the room, like I did. A serial programmable display module can easily cost $40 to $50 or more. Way too much for this project.

The key to low cost in this design is the LCD display and the way it is driven. Since only one of two four letter words are displayed, the display driver circuitry is reduced to bare essentials. Instead of using a general purpose driver or a serial interface display, two inexpensive two character seven segment LCD displays are used. Each segment has its own pin for control. In addition there is a pin to drive the back plane. The LCD display cannot be driven with DC like a LED display can. It must be driven with a 30 Hz to 300 Hz square wave. The back plane is driven with this square wave. The segments that are turned off are driven with the same square wave as the back plane. The segments that are turned on are driven with the same square wave, only 180 degrees out of phase (inverted). Since only two words, Open or Shut, are displayed I categorized the segments into four categories: Common Segments, Open Unique Segments, Shut Unique Segments, and Unused Segments. See the figure below. The Common Segments are needed to form both words and hence are on all the time. These segments are always driven with the inverse of the back plane square wave. The Open Unique Segments are needed only when the word Open is to be displayed. To display Open, these segments are driven with the inverse of the back plane square wave. When Shut is displayed, these segments are driven with the same square wave as the back plane. The Shut Unique Segments are needed only when the word Shut is to be displayed. These segments are driven similar to the Open Unique Segments. The Unused Segments are not needed to display either word, so they are always off.

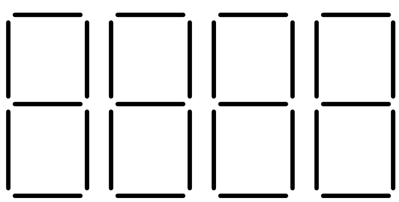
Discrete Logic Replacement

Therefore, these segments are always driven with the same square wave as the back plane. If you really want to save money, the words Open and Shut could be abbreviated as OP and SH respectfully, eliminating the second two character display.

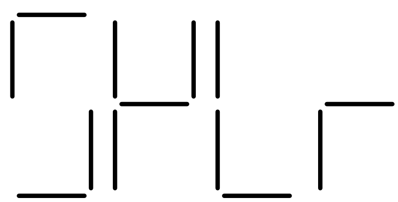
The PIC12C508 microcontroller is well adapt at generating the four square waves needed here. As you can see from the flow chart and the code listing, the lines driving the display are toggled once each time through the main loop. The timing of the main loop and hence the frequency of the square wave is accurately controlled by the PIC12C508's internal timer. At the end of each loop the code waits for the timer to hit zero before starting another loop.

Also, I was able to add a switch debounce feature to this circuit, which was not in the original circuit, at no additional cost. This was simply accomplished by setting a counter to 25 after each time the switch is read. Before the switch is read the code checks to see if the counter has reached zero. If not, it decrements the counter and skips reading the switch. Since is takes 4 mS to go through the loop, the switch will only be read every 25 * 4 mS = 100 mS. This keeps the display from flickering when the switch is opened or closed.

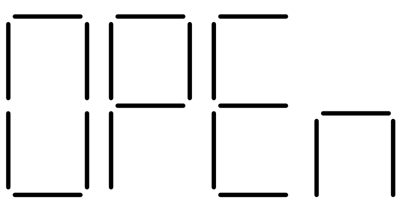
All Segments



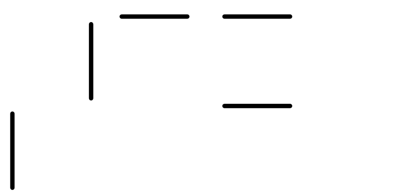
Common Segments



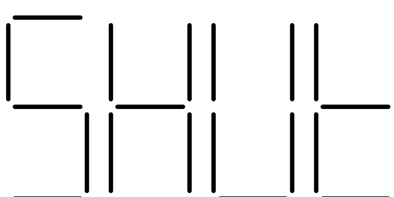
Open Segments



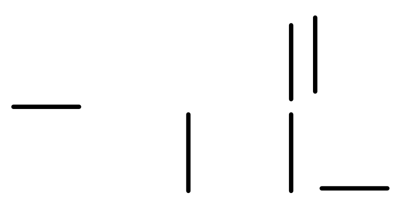
Open Unique Segments



Shut Segments

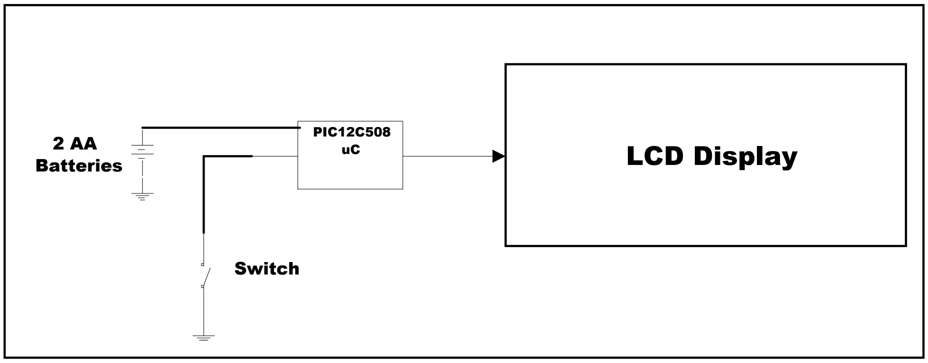


Shut Unique Segments



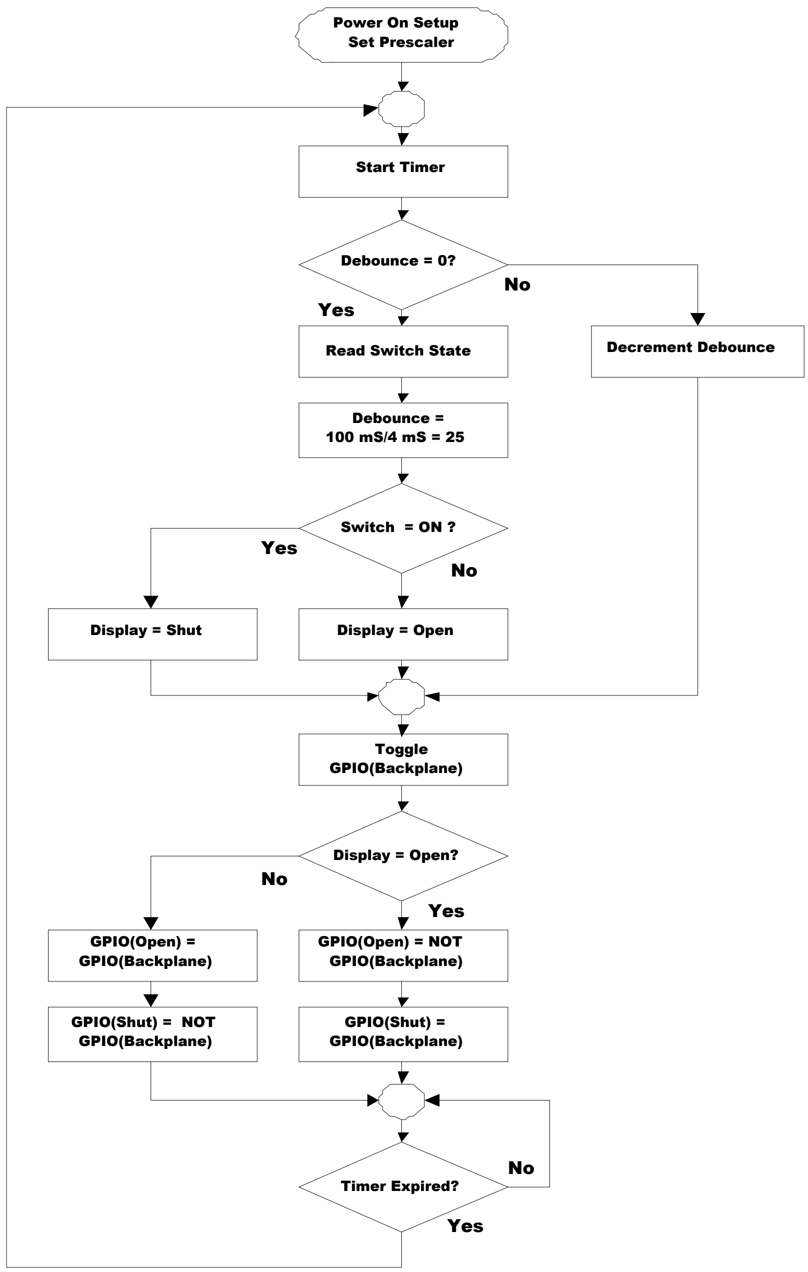
Unused Segments

Block Diagram:



Discrete Logic Replacement

Flow Chart



Graphical hardware representation:

Figure 1 shows the connections to the display for both the PIC12C508 implementation and the original discrete logic implementation. Note, the unused segments and the back plane are connected to the same pin. Figure 2 shows the PIC12C508 implementation. Figure 3 shows the original discrete logic implementation.

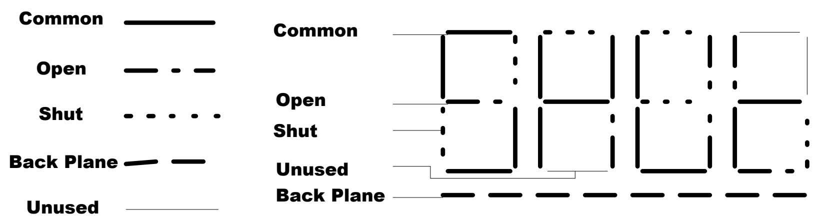


Figure 1. LCD Display Segment Conections

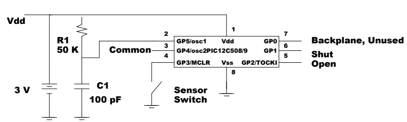


Figure 2. PIC12C508 Schematic

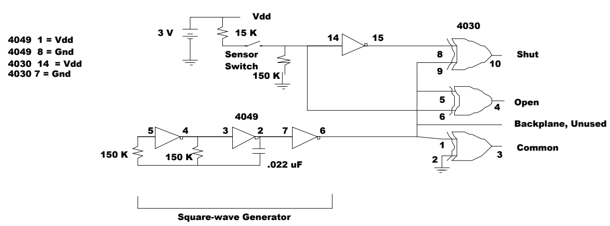


Figure 3. Discrete Logic Schematic.

Discrete Logic Replacement

Bill of Materials (BOM):

Cost is based on single quantity pricing.

PIC12C508 Implementation

Qty	Part#	Manufacture	Estimated Costs
1	PIC12C508	Microchip Technology	$1.88
1	50 K		$0.05
1	P4024A	Panasonic 100 pF	$0.07
2	VI-201-DP-RC-S	Varitronix Limited 4.30 ea	$8.60
1	SS-5GL2	Omron Roller Lever Switch	$1.66
1	BC22AAL-ND	MPD 2 cell Battery holder	$0.66

			$12.92

Original Discrete Logic Implementation

Qty	Part#	Manufacture	Estimated Costs
1	CD4030CN	National Semiconductor	$1.31
1	CD4049UBE	Harris Semiconductor	$0.54
3	150 K		$0.15
1	15 K		$0.05
1	ECQ-B1H223JF	Panasonic Polester 0.022 uF	$0.13
2	VI-201-DP-RC-S	Varitronix Limited 4.30 ea	$8.60
1	SS-5GL2	Omron Roller Lever Switch	$1.66
1	BC22AAL-ND	MPD 2 cell Battery holder	$0.66

			$13.10

APPENDIX A: SOURCE CODE

```
Title           "Garage Door / Damper Indicator"
Subtitle        "Version 1.0"

;               Written by Brian Iehl
;               Last Modified 8/23/97
;
;
                list p=12C508
;DEFINES

                INCLUDE c:\apps\mplab\p12c508.inc
SetIO           equu     B'00001000'     ;0 for output, 1 for input

GPIO0           equ      0
GPIO1           equ      1
GPIO2           equ      2
GPIO3           equ      3
GPIO4           equ      4
GPIO5           equ      5
                                         ; Outputs
BPLine          equ      GPIO0           ; Back plane and unused segments
ShutLine        equ      GPIO1           ; Shut Segments
OpenLine        equ      GPIO2           ; Open Segments
ComLine         equ      GPIO4           ; Common Segments

SWLine          equ      GPIO3           ; Switch Input
SwValue         equ      B'00001000'     ; Used to test GPIO3 bit

SHUT            equ      0               ; Switch closed so low
OPEN            equ      1               ; Switch open so high
BIT0            equ      0

ScratchPadRam   equ     0x07
Display         equ      ScratchPadRam+0        ; 0 = SHUT, 1 = OPEN
SwState         equ      ScratchPadRam+1        ; Switch State
BackPlane       equ      ScratchPadRam+2        ; Back Plane State
Debounce        equ      ScratchPadRam+3        ; Debounce time
Temp            equ      ScratchPadRam+4        ; Temporary varable


;**************************** MACROs *******************************

MOVLF  MACRO    LL,      FF              ; Move Literal to register file
       MOVLW    LL                       ; Load literal
       MOVWF    FF                       ; Store in register file
       ENDM                              ; end MOVLF


DisplayWord     MACRO                    ; Display disired word by writing to GPIO
                LOCAL Else2, EndIf2, Else3, EndIf3, Else4, EndIf4
                BTFSC    Display, BIT0   ; if Display = 0 then Display SHUT
                GOTO     Else2                  ; else
                BTFSC    BackPlane, BIT0 ; if BackPlane = 0 then
                GOTO     Else3                  ; else
                BCF      GPIO,           OpenLine; OpenLine = BackPlane
                BSF      GPIO,           ShutLine; ShutLine = NOT BackPlane
                GOTO     EndIf3                 ; else
Else3           BSF      GPIO,           OpenLine; OpenLine = BackPlane
                BCF      GPIO,           ShutLine; ShutLine = NOT BackPlane
EndIf3          GOTO     EndIf2
Else2                                           ; else Display OPEN
                BTFSC    BackPlane,      BIT0   ; if BackPlane = 0 then
                GOTO     Else4                  ; else
                BSF      GPIO,           OpenLine; OpenLine = NOT BackPlane
```

```
                BCF      GPIO,          ShutLine; ShutLine = BackPlane
                GOTO     EndIf4                 ; else
Else4           BCF      GPIO,          OpenLine; OpenLine = NOT BackPlane
                BSF      GPIO,          ShutLine; ShutLine = BackPlane
EndIf4
EndIf2
                ENDM                            ; End DisplayWord

                                                ; Generate squarewaves
Sqwave          MACRO                           ; Toggle state of backplane
                Local Else1, EndIf1

                BTFSC    BackPlane,     BIT0    ; if BackPlane = 0 then
                GOTO     Else1                  ; else
                BSF      GPIO,          BPLine  ; BPLine = 1
                BSF      BackPlane,     BIT0    ; BackPlane = 1

                BCF      GPIO,          ComLine ; ComLine = 0
                GOTO     EndIf1                 ; skip past else section

Else1                                           ; BackPlane = 1
                BCF      GPIO,          BPLine  ; BPLine = 0
                BCF      BackPlane,     BIT0    ; BackPlane = 0

                BSF      GPIO,          ComLine ; ComLine = 1

EndIf1                                          ; end if statement structure
                ENDM                            ; End Macro


;*************************** Main Program **********************************

                org      0x0A                   ;start address 0
                goto    Start
                org     0x10


;
Start


Setup           MOVLW    SetIO                  ; Load IO configuration byte
                TRIS     GPIO                   ; Set GPIO with contents of w
                CLRF     Display                ; Clear variables
                CLRF     SwState
                CLRF     BackPlane
                CLRF     Debounce
                CLRF     Temp
                                                ; Set prescaler
                                                ; Oscillator frequency = 200 kHz
                                                ; Instruction cycle = 1/(200 kHz /4) = 20 uS
                                                ; Need to generate 125 Hz square wave
wave
                                                ; So need to toggle line
(1/125)/2 = 4 mS
                MOVLW    B'10001000'            ; 4 mS / 20 uS = 200 instruction cycles
                OPTION                          ; Therefore disable prescaler timer less than
                                                  256
                                                ; This also disables wake-up on pine change
                                                ; and enables weak pull-ups

MLoop           MOVLF    D'200',        Temp    ; Main Loop must be less that 200 instructions
                                                  long
                COMF     Temp,          w       ; or need to increase osc freq.
                MOVWF    TMR0                   ; Set timer 200 * 20 uS = 4 mS
```

```
                MOVF     Debounce,        w      ; Check debounce
                BTFSS    STATUS, Z               ; Skip if zero
                GOTO     DecDebnc                ; if not don't read switch until debounced

                                                 ; Read Switch State
                MOVF     GPIO,            w      ; read GPIO register
                ANDLW    SwValue                 ; Clearall bits except SwState
                BTFSS    STATUS,          Z      ; if switch OPEN
                goto     SwSHUT                  ;
                MOVLF    OPEN,            SwState ; Display = OPEN
                goto     EndIFD

SwSHUT                                           ; else
                MOVLF    SHUT,            SwState ; Display = SHUT

                goto     EndIFD
DecDebnc
                DECF     Debounce,        f      ; Decrement Debounce and store

EndIFD
                Sqwave                           ; Toggle Backplane
                DisplayWord                      ; Set GPIO lines to Display word

                                                 ; wait for timer
WaitLp          MOVF     TMR0,            w      ; force check zero
                BTFSS    STATUS,          Z      ; timer expired?  w = 0 if done, so Z is set
                goto     WaitLp                  ; not 0 so loop again
                                                 ; one more mS passed

                GOTO     MLoop                   ; loop again

                END
```


NOTES: