APPLICATION OPERATION

The electronic switch discussed in this application note may be used to operate five different appliances. It is based on the PIC12C508 and requires few external components. The program has a built in interpreter with loops which can output data to PICmicro pins. The language used in the interpreter has seven commands. The switch has five buttons to operate five outputs, one line-in terminal, and five line-out terminals.

With a few external components added, you can make Christmas lights for a Christmas tree. Attach five horns to your car, and they will play your favorite melodies. It is popular to animate car brake lights now. If connected to a timer via ‘Line In’, this switch can manipulate five appliances. Switches can be connected together through their ‘Line Out’ and ‘Line In’ terminals. This will increase the number of connected appliances.

FIGURE 1: SCHEMATIC DIAGRAM

![Schematic Diagram]

Note: Each Line In and Line Out terminal must have its own grounding terminal.
THE LIGHTS PARTS LIST

Capacitors:
C1 - 47 µF electrolytic

Diodes:
D1-D6 - Any type diodes
D7-D11 - Any type red light emitting diodes (LEDs may be replaced with Opto-relays or opto-triacs. See Programmable Timer with Time Correction on how to connect to outlet. Special care should be taken when a transformerless power supply is used (see TB008) because any damage to grounding circuit will destroy ICs if two or more Programmable Lights are connected together.)

Resistors:
R1-R5 – depends on the type of LED
VO = PIC12C508 output low voltage (0.6V Max)
VLED = Input LED voltage (may vary from 0.8V IR to 2.0V green LED) (1.5 V red LED)
ILED = LED current (10 mA Typical)

R1 = \frac{5\text{ - } VLED - VO}{ILED} = 290\Omega

Miscellaneous
S1-S5 - normally open pushbutton switches
U1 - PIC12C508 programmed with Lights code

SOFTWARE

The program outputs the value of the Light command every 16 ms.

How the Keyboard Works.

First, all outputs (GP5, GP4, GP2, GP1, GP0) are raised high. Then, the first output GP0 (connected to button S1) is driven low (for 5 µs). On the third microsecond S1 is sampled. If the button is pressed, then GP3 is low. Each button has two key flags. Buttons are tested every 16 ms. If button is not pressed, then the flag is cleared. When the button is pressed and the flag is cleared, it is raised and the program continues to output lights. If the flag is raised already, then a required function is activated, and the second flag is raised else it is assumed to be a key debounce and flags are cleared. All buttons are tested sequentially. A low signal on the ‘Line In’ terminal will animate button S1 depression (this may be changed in code). To prevent reading buttons on polling of remote Programmable Lights connected to ‘Line In’, an additional check is made when all pins are high (this is needed if two appliances are absolutely synchronized).

PROGRAMMABLE LIGHTS MACRO LANGUAGE

Macro language used in this appliance has 7 commands. They are summarized in Table 1.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Light</td>
<td>Set new light</td>
</tr>
<tr>
<td>MarkLabel</td>
<td>Mark a label</td>
</tr>
<tr>
<td>ReturnToLabel</td>
<td>Return to label</td>
</tr>
<tr>
<td>SetRepeatValue</td>
<td>Set repeat value</td>
</tr>
<tr>
<td>JumpToProgram</td>
<td>Jump to program</td>
</tr>
<tr>
<td>StartOfProgram</td>
<td>Start of program</td>
</tr>
<tr>
<td>RestartProgram</td>
<td>Return to start of program</td>
</tr>
</tbody>
</table>

Light

This command is used to set microcontroller pins high or low. It is advisable to write this command like this:

Light b’000001’

Binary data sent to PICmicro pins is (from right to left): GP5, GP4, GP2, GP1, GP0, where 0 sets the corresponding pin high, and 1 sets low. The command shown in the example will turn GP5, GP4, GP2 and GP1 high, and GP0 low.

SetRepeatValue MarkLabel ReturnToLabel

These commands are used to make a loop with counter in the program. SetRepeatValue should precede MarkLabel command and is used to set the number of times the program will when ReturnToLabel command found roll over to MarkLabel. There is no default value, so each MarkLabel command must have a SetRepeatValue command. There may be 31 labels in the program. When a ReturnToLabel command is found the interpreter decreases repeat value and if it is not zero than it walks backward (this is significant, because in this case there may be more than 31 loops in the program) until a MarkLabel command is found. The repeat value should be in the range from 1 to 31 (if SetRepeatValue is set to 0 then commands inside loop will be output 256 times). An example shows how to write a loop correctly:

```
SetRepeatValue 10
MarkLabel 2
Light b’00001’
ReturnToLabel 2
```

This code example sets repeat value to 10 times and uses label #2. This code will set pin GP0 low ten times longer than the basic delay. Label numbers make no sense now but with some improvement to the code some label numbers may have unique counters and loops may be nested.
Start of Program RestartProgram

Part of program may be placed into an endless loop. To switch between endless loops this appliance has three buttons: Reset, Test and Program. There may be 31 loops. An example shows how it works:

```
StartOfProgram 5
.
.
StartOfProgram 1
Light b’00001’
RestartProgram 5
```

If the current program number is #1 then command RestartProgram 5 is ignored. When RestartProgram 1 is fetched the interpreter walks forward until StartOfProgram 1 is found. Programs may be nested or crossed.

This commands may be useful to skip some code when several programs are nested or crossed. This is shown in the example:

```
StartOfProgram 1
Light b’00001’
RestartProgram 1
Light b’00100’
StartOfProgram 1
Light b’00010’
RestartProgram 1
```

In this case if the current program number is 1 then the interpreter will output 00001 and 00010 skipping 00100. This is because when RestartProgram is found the interpreter walks forward until the desired StartOfProgram is found.

Jump to Program

This command will jump to the desired StartOfProgram command. An example shows how it works:

```
.
.
.
StartOfProgram 1
Light b’00001’
RestartProgram 1
.
.
.
JumpToProgram 1
.
.
```

When JumpToProgram 1 is fetched the interpreter walks forward until StartOfProgram 1 is found. It is not the same as with RestartProgram since current program number may be not #1 when JumpToProgram 1 is fetched.

With the help of this command a simple alarm system may be done. System block diagram is shown in figure 2. Look at the code:

```
define MAX_PROGRAM=1
StartOfProgram Test
Lights b’00000’
RestartProgram Test

StartOfProgram 1
SetRepeatValue 10
MarkLabel 1
Lights b’10001’
Lights b’10000’
ReturnToLabel 1
JumpToProgram Test
```
Line in is connected to a car door open switch (normally closed contacts with one terminal connected to the ground) through a rectifier. GP0 is connected through an amplifier and a relay to a horn. GP5 is connected to line in. Other outputs maybe connected to parking lights and ignition immobilizer. Once the door is open this will change the current program number to #1 and the horn will beep 10 times. Line-in will be blocked with GP5 low signal to prevent several restarting of program number 1. Then the program will jump to test program and line in will be connected again. If the door is still opened there will be another beep. To switch the alarm off you need to push a Test button. If the thief tries to switch the car battery off and then on again the program will automatically start from program #1 and an alarm will occur.

QUICK CODE IDEAS

The code has a macro to test pressed buttons and a small function to read tables.

A macro to test pressed buttons is useful when the microcontroller has no interrupt flag on pin change and the only way to add a keyboard is to poll microcontrollers pins. This macro needs two bytes to store key flags (for 8 buttons or less).

I know several table read functions. Some are simple the rest are too long. This function is ideal for PIC12C508/509 since a call may be done only for the first 256 bytes of memory page code.

MICROCHIP TOOLS USED

Development Tools:

This code was written and debugged with MPLAB™ for Windows®/16 Version 3.22.02.

Assembler/Compiler version:

MPASM v01.50
APPENDIX A: SOURCE CODE

;Programmable Lights
;Author: Kirill Yelizarov

LIST P=PIC12C508, R=DEC
INCLUDE <pl12c508.inc>

__CONFIG _IntRC_OSC & _WDT_OFF & _CP_OFF & _MCLRE_OFF

; -------------------- DATA --------------------
MPC   equ 0x07 ;Macro program counter
KeyFlags equ 0x08 ;Keyboard flags
Keys equ 0x09 ;Keyboard keys
Command equ 0x0a ;Current command
Value equ 0x0b ;Local data
Program equ 0x0c ;Current program number
TimeDelay equ 0x0d ;Time delay changed with Inc or Dec buttons
TimeCount equ 0x0e ;Current time count
RepeatCount equ 0x0f ;Counter for a loop

;----- Keyboard bits ----- 
ResetKey equ 5 ;Switch to program #1
TestKey equ 4 ;Switch to program #0 (called Test)
LineIn equ 3 ;Line in for all buttons
DecKey equ 2 ;Decrease delay between lights flow
IncKey equ 1 ;Increase delay between lights flow
ProgKey equ 0 ;Switch to next program

; -------------------- Lights macro language --------------------
#define Light retlw 0x00+ ;Set new light
#define MarkLabel retlw 0x20+ ;Mark a label #(1...31)
#define ReturnToLabel retlw 0x40+ ;Return to label #(1...31)
#define SetRepeatValue retlw 0x80+ ;Set repeat value (1...31)
#define JumpToProgram retlw 0xa0+ ;Jump to program #(1...31)
#define StartOfProgram retlw 0xc0+ ;Start of program #(1...31)
#define RestartProgram retlw 0xe0+ ;Return to start of program #(1...31)

Mark equ 0x20
ReturnTo equ 0x40
RepeatValue equ 0x80
JumpTo equ 0xa0
StartOf equ 0xc0
Restart equ 0xe0

Button equ 0x00
Line equ 0x01
MAX_PROGRAM equ 12
Test equ 0 ;Test program

;-------------------- MACRO --------------------
;This macro is used to search pressed button
;total button de bounce test is 2*16 ms = 32 ms with one extra test on the 16th ms

TestButtonmacro source,line,butt,key,flag,out
local reset
local test
Electromechanical Switch Replacement

bcf GPIO, butt ; drive butt pin low
nop ; wait
btfsc GPIO, line ; look what happens on line pin
goto reset ; if high go and reset all for this button
bsf GPIO, butt ; drive butt pin high

IF source==Button ; extra test for synchronized buttons
nop ; wait
btfss GPIO, line ; test pin for high value
goto reset ; if pin is still low then it's a line in signal
ENDIF

IF source==Line
nop ; wait
btfsc GPIO, line ; test line in if signal is still low
goto reset ; else a polling signal was read
ENDIF

btfsc key, butt ; test key depression for butt pin
goto test ; yes there was a depression on last test, go to test
bsf key, butt ; no then set key depression flag
goto out ; and wait for the next test

reset:
bsf GPIO, butt ; drive butt pin high
bcf key, butt ; clear depression key
bcf flag, butt ; clear key flag
goto out ; test next
test:
btfsc flag, butt ; test key flag
goto out ; if set then test next (to prevent multiple run of the
; same function while the button is pressed)
bsf flag, butt ; set key flag
; function code will be merged here
endm

;----------------------- CODE -----------------------
org 0
goto Start ; Skip subroutine and table

; This multifunction subroutine can fetch next, or previous, or current command from the table

DecPC ; Get previous command
decf MPC, F
movlw low Table
xorwf MPC, W
btfss STATUS, Z
goto ReadPC
movlw low (Start-1)
movwf MPC
goto ReadPC

IncPC ; Get next command
incf MPC, F
movlw low Start
xorwf MPC, W
btfss STATUS, Z
goto ReadPC
movlw low (Table+1)
movwf MPC
ReadPC ; Read current command
movf MPC, W
call Table
movwf Value
retlw 0
; ------------------- P R O G R A M ---------------------
Table
movwf PCL
; ***********************************************
; * Christmas Lights *
; * 1997 *
; * Author: Kirill Yelizarov *
; ***********************************************

StartOfProgram Test
Light b'11111'
RestartProgram Test

StartOfProgram 1
Light b'11111'
Light b'11111'
Light b'11111'
Light b'01111'
Light b'01111'
Light b'01111'
Light b'00111'
Light b'00111'
Light b'00111'
Light b'00111'
Light b'00111'
Light b'00011'
Light b'00011'
Light b'00011'
Light b'00011'
SetRepeatValue 10
MarkLabel 1
StartOfProgram 2
Light b'10000'
Light b'01000'
Light b'00100'
Light b'00010'
Light b'00001'
RestartProgram 2
ReturnToLabel 1
SetRepeatValue 10
MarkLabel 2
StartOfProgram 3
Light b'10001'
Light b'11000'
Light b'01100'
Light b'00110'
Light b'00011'
RestartProgram 3
ReturnToLabel 2
SetRepeatValue 10
MarkLabel 3
StartOfProgram 4
Light b'10011'
Light b'11001'
Light b'11100'
Light b'01110'
Light b'00111'
RestartProgram 4
ReturnToLabel 3
SetRepeatValue 10
MarkLabel 4
StartOfProgram 5
Light b'10111'
Light b'11011'
Light b'11101'
Light b'11110'
Light b'01111'
RestartProgram 5
ReturnToLabel 4

SetRepeatValue 10
MarkLabel 5
Light b'00111'
Light b'10011'
Light b'11001'
Light b'11100'
Light b'01110'
ReturnToLabel 5

SetRepeatValue 10
MarkLabel 6
Light b'00110'
Light b'00011'
Light b'10001'
Light b'11000'
Light b'01100'
ReturnToLabel 6

SetRepeatValue 10
MarkLabel 7
Light b'00100'
Light b'00010'
Light b'00001'
Light b'10000'
Light b'01000'
ReturnToLabel 7

Light b'00100'
Light b'00010'
Light b'00001'
StartOfProgram 6
Light b'10000'
Light b'10000'
Light b'10000'
Light b'11000'
Light b'11000'
Light b'11000'
Light b'11100'
Light b'11100'
Light b'11100'
Light b'11110'
Light b'11110'
Light b'11110'
Light b'11100'
Light b'11100'
Light b'11100'
Light b'11100'
Light b'11000'
Light b'11000'
Light b'11000'
Light b'10000'
Light b'10000'
Light b'10000'
Light b'10000'
Light b'10000'
RestartProgram 6
SetRepeatValue 10
MarkLabel 8
StartOfProgram 7
Light b'00001'
Light b'00010'
Light b'00100'
Light b'01000'
Light b'10000'
RestartProgram 7
ReturnToLabel 8
SetRepeatValue 10
MarkLabel 9
StartOfProgram 8
Light b'100001'
Light b'00011'
Light b'00110'
Light b'01100'
Light b'11000'
RestartProgram 8
ReturnToLabel 9
SetRepeatValue 10
MarkLabel 10
StartOfProgram 9
Light b'11001'
Light b'10011'
Light b'00111'
Light b'01110'
Light b'11100'
RestartProgram 9
ReturnToLabel 10
SetRepeatValue 10
MarkLabel 11
StartOfProgram 10
Light b'11101'
Light b'11011'
Light b'10111'
Light b'01111'
Light b'11110'
RestartProgram 10
ReturnToLabel 11
SetRepeatValue 10
MarkLabel 12
Light b'11100'
Light b'11001'
Light b'10011'
Light b'00111'
Light b'01110'
ReturnToLabel 12
SetRepeatValue 10
MarkLabel 13
Light b'01100'
Light b'11000'
Light b'10001'
Light b'00011'
Light b'00110'
ReturnToLabel 13

SetRepeatValue 10
MarkLabel 14
Light b'00100'
Light b'01000'
Light b'10000'
Light b'00001'
Light b'00010'
ReturnToLabel 14

Light b'00100'
Light b'01000'
Light b'10000'

StartOfProgram 11
Light b'00001'
Light b'00001'
Light b'00001'
Light b'00011'
Light b'00011'
Light b'00011'
Light b'00111'
Light b'00111'
Light b'00111'
Light b'00111'
Light b'00111'
Light b'00111'
Light b'01111'
Light b'01111'
Light b'01111'
Light b'01111'
Light b'01111'
Light b'01111'
Light b'01111'
Light b'01111'
Light b'01111'
Light b'01111'
Light b'01111'
Light b'01111'
Light b'01111'
Light b'00111'
Light b'00111'
Light b'00111'
Light b'00111'
Light b'00111'
Light b'00111'

RestartProgram 1

StartOfProgram 12
Light b'10001'
Light b'00000'
Light b'01010'
Light b'00000'
Light b'00100'
Light b'00000'
Light b'01010'
Light b'00000'
Light b'00000'
Light b'00000'
Light b'00000'
Light b'00000'
Light b'00000'
Light b'00000'
Light b'00000'
Light b'00000'
Light b'00000'
Light b'00000'
Light b'00000'
Light b'00000'
Light b'00000'
Light b'00000'

; -------------- M A I N --------------
Start:

IF Start>0x100
ERROR "Lights Message: Program Table too large."
ENDIF

clf Keys ;Reset key depression
clrf KeyFlags ;Reset keyboard flags
cld ;clear TMRO
movlw b'10000101' ;Enable weak pull-up on GP3 and set prescaler to 1:64
option
movlw b'10000000' ;*****For MPLAB debug
clrf GPIO
comf GPIO,F ;Turn OFF the lights
movlw b'00001000' ;Set GP3 (LineIn) as input and the rest are outputs
tris GPIO
movlw low (Start-1) ;Set program counter to the end of Table
movwf MPC
movlw 0x10
movlw 0x01
movwf IncPC ;Each time TMRO overflows this value will decrement
movwf TimeDelay ;Set time dalay between two Light commands to 16*16 ms
= 256 ms may be changed with Inc and Dec keys
movlw 0x01
movwf Program ;Set program #1

SetProgram:
call IncPC ;Get next command
movf Value,W
andlw b'11100000'
xorlw StartOf ;If it's a StartOfProgram then continue
btfss STATUS,Z ;else go and get another command
goto SetProgram
movf Value,W
andlw b'00011111'
andwf Command,W ;Save Command because it's contents are no longer needed
movf Value,W
andlw b'00011111'
andwf Command,W ;Save label in Command because it's contents are no longer needed
decfsz RepeatCount,F ;decrease loop counter set by
goto MainLoop
FindLabel:
call IncPC ;Search backward for a desired label
movf Value,W
andlw b'11100000'
xorlw Mark
btfss STATUS,Z
goto FindLabel
movf Value,W
andlw b'00011111'
andwf Command,W
goto MainLoop

MainLoop:
call IncPC ;Get next command
movf Value,W
andlw b'11100000'
btfsc STATUS,Z ;Test Light Command
goto SetLights
movf Value,W
andlw b'00011111'
andwf Command,W ;Save Command needed
decfsz RepeatCount,F ;decrease loop counter set by
goto MainLoop
FindLabel:
TestRestartProgram: ;Look if it's a RestartProgram command
  movf Command,W
  xorlw Restart
  btfss STATUS,Z
  goto TestRepeatValue
  movlw b'00011111'
  andwf Value,W
  xorwf Program,W
  btfss STATUS,Z
  goto MainLoop
  goto SetProgram

TestRepeatValue:
  movf Command,W
  xorlw RepeatValue
  btfss STATUS,Z
  goto TestJumpTo
  movlw b'00011111'
  andwf Value,W
  movwf RepeatCount
  goto MainLoop

TestJumpTo:
  movf Command,W
  xorlw JumpTo
  btfss STATUS,Z
  goto MainLoop
  movlw b'00011111'
  andwf Value,W
  movwf Program
  goto SetProgram

SetLights:
  movlw b'00110111'
  btfsc Value,4
  bsf Value,5
  bcf Value,4
  btfsc Value,3
  bsf Value,4
  andwf Value,F ;cut everything but the lights

Wait1:
  btfss TMR0,7 ;wait till seventh bit rise, comment it when in MPLAB debug
  goto Wait1 ;***** Comment it when in MPLAB debug
  movlw b'11111111'
  movwf GPIO

TestButton Button,LineIn,ProgKey,Keys,KeyFlags,TestInc

NextProgram:
  incf Program,F
  movlw MAX_PROGRAM+1
  xorwf Program,W
  btfss STATUS,Z
  goto SetProgram
  movlw 0x01
  movwf Program
  goto SetProgram

TestInc:
  TestButton Button,LineIn,IncKey,Keys,KeyFlags,TestDec
  movlw 0x01
  movwf TimeCount ;Set TimeCount to one to make delay change fast
  incfsz TimeDelay,F
  goto TestDec
movlw 0xff
movwf TimeDelay

testDec:
  TestButton Button, LineIn, DecKey, Keys, KeyFlags, TestLine
  movlw 0x01
  movwf TimeCount ; Set TimeCount to one to make delay change fast
  decfsz TimeDelay, F
  goto TestLine
  movlw 0x01
  movwf TimeDelay
  goto TestLine

TestLine:
  TestButton Line, LineIn, LineIn, Keys, KeyFlags, TestTest
  goto NextProgram

TestTest:
  TestButton Button, LineIn, TestKey, Keys, KeyFlags, TestReset
  clrf Program
  goto SetProgram

TestReset:
  TestButton Button, LineIn, ResetKey, Keys, KeyFlags, RestoreLights
  goto Start

RestoreLights:
  comf Value, W
  movwf GPIO ; Output Lights to GPIO

Wait0:
  btfsc TMRO, 7 ; Wait till the seventh bit reset, *****Comment it when in
  goto Wait0 ; MPLAB Debug
  decfsz TimeCount, F
  goto Wait1 ; If zero not reached then make another key check
  movf TimeDelay, W ; Reset TimeCount with TimeDelay value
  movwf TimeCount
  goto MainLoop

org 0x1ff
  movlw b'01110000' ; Set OSCCAL
end