APPLICATION OPERATION

The design is for FM tuner applications. It is a simple circuit that can be easily attached to any FM tuner so that pushbutton tuning is achieved and at the same time provides a few memories to store and recall your favorite channels on FM broadcast band. There are pushbuttons for user control. The functions are explained in Table 1.

The keyboard is very easy to operate. Stations can be tuned by momentarily pressing the up or down tuning button. The receiver locks in to the next station. If any of these buttons are pressed continuously, then continuous tuning is achieved. Similarly stored stations can be recalled at any time by just pressing the required memory button. The heart of the circuit is the PICmicro™ microcontroller which does two major functions:

- Generating PWM
- Keyboard scanning

Upon power-up, the PICmicro™ microcontroller loads with the PWM default value from ROM and produces a PWM wave at the output. Even though a high frequency PWM is preferred, this program generates PWM with a period of 255 ms. This PWM is converted into DC voltage by integrating. A simple RC integrator will be sufficient for this purpose. The upper cut-off frequency is set to a few Hz. The PWM wave is buffered and level converted by the NPN transistor, T1. It is a general-purpose NPN transistor such as BC547 or 2N2222, etc. Receiver tuning is achieved by giving this voltage to the local oscillator. The local oscillator should be a VCO-type. Tuning voltage varies the junction capacitance of the varactor diode in the oscillator tank circuit and, hence, the frequency. By varying the Pulse width of the PWM wave form, the tuning voltage varies and hence tuning. It is important to note that in low-cost receivers, the front end RF amplifier is of wide band type, avoiding any tuning needed. But, in high sensitive receivers, the front end must also be tuned in accordance with the Local Oscillator (LO). This can be achieved by making the front end also varactor tuned. Receiver front tuning can thus be done by deriving the control voltage from tuning voltage to LO. This arrangement is not shown in the schematic.

A feedback from the receiver is given to GP3 of the PICmicro™. This input goes high when a station of sufficient strength is tuned. Carrier detect is a logic signal derived from the AGC voltage developed in the receiver IF stage. A simple comparator, as shown in the Block Diagram on page 2, can be used to get the carrier detect signal. The carrier threshold or signal strength can be adjusted by the potentiometer (P1). During tuning, if a carrier detect signal is received by the PICmicro™, the present value in the PWM register is held constant and, thereby, maintains a constant DC voltage as the tuning voltage. An automatic frequency control loop in the receiver circuitry fine tunes the LO to receive the station. Now the PWM value is changed again only if tuning button connected to PICmicro™ microcontroller is activated again. PWM values corresponding to any station can be stored and recalled by appropriate action of the memory switches. In this circuit three memories are shown, which is sufficient for most purposes. More memories can added if desired.

For interfacing the function switches (all are non-locking pushbuttons) a diode matrix is used. The advantages are low power consumption, low cost, and less space needed for PCB. With four lines to the PICmicro™ from the matrix, it can have a maximum of 15 switches. Here we use only 6.

TABLE 1: PUSH-BUTTON FUNCTIONS

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>UP TUNE</td>
<td>When pressed, the receiver tunes to the next higher frequency side station.</td>
</tr>
<tr>
<td>DOWN TUNE</td>
<td>When pressed, receiver tunes to the next lower frequency station.</td>
</tr>
<tr>
<td>STORE</td>
<td>press this button to store the current tuned frequency to memory specified by the next button to be pressed.</td>
</tr>
</tbody>
</table>
Generating PWM

The most complex part of the program is the generation of the PWM waveform. The waveform should be accurate to produce good results. The clock does not need to be very stable, as the AFC loop in the receiver always takes care of the small drift in the LO frequency due to instability of the PICmicro clock or parameter change of the frequency determining elements in the local oscillator circuit. Hence, the internal RC oscillator will be sufficient.

The TMR0 is used to generate a fairly good PWM wave, without sacrificing other routines needed for the software.

### TABLE 1: PUSH-BUTTON FUNCTIONS

| M1, M2, M3 | These are the memory buttons Stations can be memorized in any of these by pressing the store button followed by the M1, M2, and M3. |

### BLOCK DIAGRAM

![Circuit Diagram of PIC12C5XX based FM Tuner Controller](image)

### TABLE 2: KEYBOARD MAPPING

<table>
<thead>
<tr>
<th>Keys</th>
<th>Address Read By PICmicro™ Microcontroller</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tune Up</td>
<td>X1110</td>
</tr>
<tr>
<td>Tune Down</td>
<td>X1101</td>
</tr>
<tr>
<td>Store</td>
<td>X1100</td>
</tr>
<tr>
<td>M1</td>
<td>X1011</td>
</tr>
<tr>
<td>M2</td>
<td>X0111</td>
</tr>
<tr>
<td>M3</td>
<td>X0011</td>
</tr>
</tbody>
</table>
SOFTWARE LISTING

;-----------------------------------------------------
;This program runs on PIC12C5XX
;This program is written as control program
;for FM tuner
;WRITTEN BY MANI.T.K(VU2ITI)
;
;-----------------------------------------------------

register equ 0x07
M1   equ register+0
M2   equ register+1
M3   equ register+2
PWM  EQU register+3
DELAY EQU register+4
STATUS equ 0x03
output equ 05
w     equ 0
f     equ 1
GPIO  EQU 06
TMR0  EQU 01

; org 0 ; start address 0

start
    movlw 0x81 ; Initialise timer 0
    option
    movlw 0x1f
    tris GPIO ; configure the input and output
    movlw 0x7f
    movwf TMR0
    BSF GPIO,output ; set output

; BEGIN

    MOVF TMR0,f
    BTFSC STATUS,02 ; check for timeout

; TIMOUT
    CALL UPDATE ; If timeout, call update
    MOVF GPIO,w ; read input
    ANDLW 0X0C ; check for m3
    BTFSO TMR0,f
    GOTO M3TOPWM ; if m3, m3 to pwm

    MOVF TMR0,f
    BTFSC STATUS,02 ; check for timeout
    CALL UPDATE ; If timeout, call update
    MOVF GPIO,w ; read input
    ANDLW 0X08 ; check for m2
    BTFSO TMR0,f
    GOTO M2TOPWM ; if m2, m2 to pwm

    MOVF TMR0,f
    BTFSO STATUS,02 ; check for timeout
    CALL UPDATE ; If timeout, call update
    MOVF GPIO,w ; read input
    ANDLW 0x04 ; check for m1
    BTFSO STATUS,02
    GOTO M1TOPWM ; if m1, m1 to pwm

; STORE
    MOVF GPIO,w
    ANDLW 0x13 ; Check is store button is pressed
    BTFSO STATUS,02 ; if store, goto STORE
TUNE ;Tuning routine.

MOVF GPIO,w ;read input
ANDLW 0x01 ; if tune up, inc pwm
BTFSC STATUS,02 ; if tune up, inc pwm
INCF PWM,F
MOVF GPIO,w ;read input
ANDLW 02
BTFSC STATUS,02 ;if tune down dec pwm
DECF PWM,F
MOVF PWM,W
MOVWF TMR0
MOVF TMR0,f
BTFSC STATUS,02 ;check for timeout
CALL UPDATE ;If timeout, call update
BTFSC GPIO,05 ;Check for carrier detect
GOTO TUNE ; If no carrier, Continue tuning
GOTO BIGIN

;----------------------------------UPDATE Souboutine----------------------------------

UPDATE

COMF GPIO,w ; compliment output
MOVF PWM,f
MOVF PWM,w
MOVF TMR0 ;pwm to TMR0 And compliment
MOVWF TMR0,f
RETLW 0x00 ;return from subroutine

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

; routines for recalling memory1

M1TOPWM

MOVF M1,W
MOVWF PWM
GOTO BIGIN

; routines for recalling memory2

M2TOPWM

MOVF M2,W
MOVWF PWM
GOTO BIGIN

; routines for recalling memory

M3TOPWM

MOVF M3,W
MOVWF PWM
GOTO BIGIN

; this can be extended further for more memories.

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

STORE

MOVWF DELAY
MOVWF PWM

LOOP

MOVF GPIO,w ;read input
ANDLW 0x0C ;check for m3
BTFSC STATUS,02 ;if m3, pwm to m3
MOVF TMR0,f
BTFSC STATUS,02 ;check for timeout

CALL UPDATE ;If timeout, call update
MOVF GPIO,w ;read input
ANDLW 0x08 ;check for m2
BTFSC STATUS,02 ;if m2, pwm to M2
GOTO PWMTOM2
; check for timeout
MOVF TMR0, f
BTFSC STATUS, 02
CALL UPDATE
MOVF GPIO, w
ANDLW 0x04
BTFSC STATUS, 02
GOTO PWMTOM1
DECF SFZ DELAY, F
GOTO LOOP
GOTO BIGIN

; routines for storing memory
PWMTOM1
MOVF PWM, W
MOVF M1
GOTO BIGIN

PWMTOM2
MOVF PWM, W
MOVF M2
GOTO BIGIN

PWMTOM3
MOVF PWM, W
MOVF M3
GOTO BIGIN

END
NOTES: