599 Menlo Drive, Suite 100
Rocklin, California 95765, USA
Office: (916) 624-8333
Fax: (916) 624-8003

General: info@parallax.com
Technical: support@parallax.com
Web Site: www.parallax.com
Educational: www.stampsinclass.com

Parallax Audio Amplifier AppMod (#29143)

Introduction

The BASIC Stamp® has two commands that generate sounds: **DTMFOUT** and **FREQOUT**. To make the most of these commands, a filter circuit and audio amplifier should be used. The Audio Amplifier AppMod makes it easy and is a great way to add sound output to your BOE-Bot, Toddler, or other BASIC Stamp-based project.

Features

- 125 milliwatt amplifier
- Adjustable volume control
- Built-in speaker
- Selectable external speaker connection
- Parallax standard AppMod format – plugs right in to a Board of Education (BOE)

Packing List

The Parallax Audio Amplifier AppMod (#29143) package should include the following components (demo source code can be downloaded from www.parallax.com):

- Documentation (these pages)
- Audio Amplifier AppMod module
- 2 x 10 female-male connector (AppMod extender)
- ¾" male-female hex stand-off, 4-40 x $^{3}/_{8}$" screw and nut

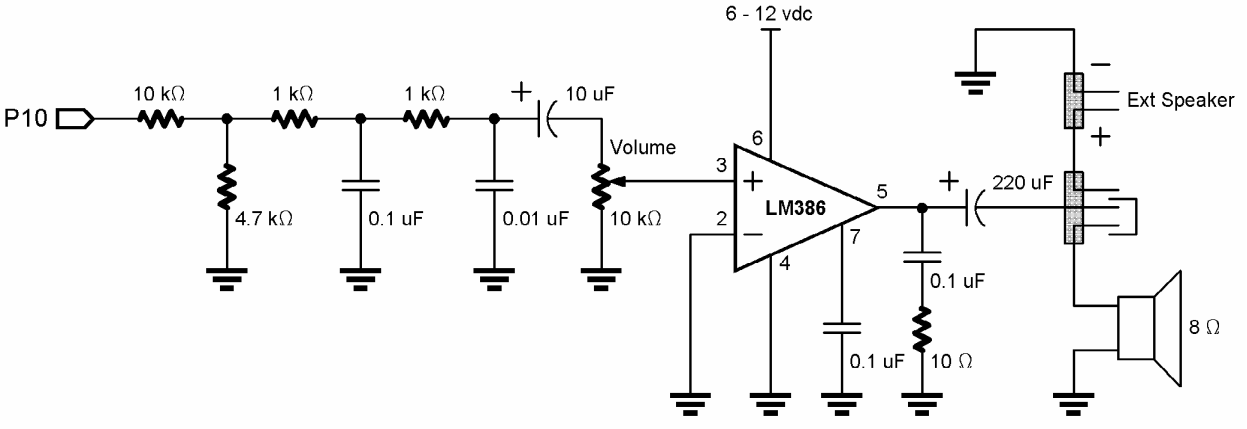
Setting Up

Follow these steps to install the Audio Amplifier AppMod on a Board of Education (Rev B or later):

- Verify that no power is connected to the Board of Education (power switch off on Rev C board)
- Plug the 2 x 10 AppMod extender into the X1 connector; making sure all pins are properly aligned
- Plug the Audio Amplifier AppMod into the 2 x 10 AppMod extender; making sure all pins are properly aligned
- Using the stand-off, screw and nut, secure the Audio Amplifier AppMod to the Board of Education

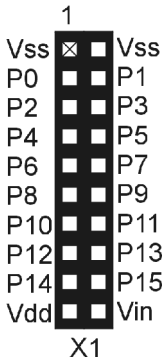
Description

The Audio Amplifier AppMod employs an industry-standard LM386 audio amplifier. Raw audio output is taken from BASIC Stamp pin P10 and filtered prior to amplification; this converts the pulse-width modulation output of the BASIC Stamp to sinusoidal waves for the best audio quality. The amplified signal is fed to an onboard 8-ohm speaker. You may connect an external speaker for improved sound quality and volume level.



AppMod Connection Pinout

You may wish to incorporate the Audio Amplifier AppMod into a PCB of your own design. When this is the case, you must ensure that your connections correspond to the standard Parallax AppMod pinout as shown in the diagram below.



Demonstration Program

The demonstration program illustrates the wide variety of possible sound effects using **DTMFOUT** and, especially, **FREQOUT**. Using conditional compilation techniques available with PBASIC 2.5 syntax, this program will run and sound the same on any BS2-family module.

```
' ============================================================================
'
'   File...... STAMP_SOUNDS.BS2
'   Purpose... BASIC Stamp Sound Effects Demo
'   Author.... Parallax, Inc. – Copyright 2002, All Rights Reserved
'   E-mail.... support@parallax.com
'   Started...
'   Updated... 17 AUG 2004
'
'   {$STAMP BS2}
'   {$PBASIC 2.5}
'
' ============================================================================


' -----[ Program Description ]----------------------------------------------
'
' This program demonstrates the versatility of the BS2 FREQOUT command.
' These sounds are best played through an audio amplifier like the Parallax
' Audio Amplifier AppMod (#29143).
'
' This program was written by Parallax with the contributions of several
' [very creative] BASIC Stamp users.


' -----[ Revision History ]-------------------------------------------------
'
' 17 AUG 2004 - Update to PBASIC 2.5 so that program behaves the same way
'               on any BASIC Stamp module.


' -----[ I/O Definitions ]--------------------------------------------------

Spkr            PIN     10                      ' Speaker / amplifier pin


' -----[ Constants ]-------------------------------------------------------

#SELECT $STAMP
  #CASE BS2, BS2E
    TmAdj       CON     $100                    ' x 1.0 (time adjust)
    FrAdj       CON     $100                    ' x 1.0 (freq adjust)
  #CASE BS2SX
    TmAdj       CON     $280                    ' x 2.5
    FrAdj       CON     $066                    ' x 0.4
  #CASE BS2P
    TmAdj       CON     $3C5                    ' x 3.77
    FrAdj       CON     $044                    ' x 0.265
  #CASE BS2PE
```

```
    TmAdj       CON     $100                    ' x 1.0
    FrAdj       CON     $0AA                    ' x 0.665
#ENDSELECT


R               CON     0                       ' rest
C               CON     33                      ' ideal is 32.703
Cs              CON     35                      ' ideal is 34.648
D               CON     37                      ' ideal is 36.708
Ds              CON     39                      ' ideal is 38.891
E               CON     41                      ' ideal is 41.203
F               CON     44                      ' ideal is 43.654
Fs              CON     46                      ' ideal is 46.249
G               CON     49                      ' ideal is 48.999
Gs              CON     52                      ' ideal is 51.913
A               CON     55                      ' ideal is 55.000
As              CON     58                      ' ideal is 58.270
B               CON     62                      ' ideal is 61.735

N1              CON     500                     ' whole note
N2              CON     N1/2                    ' half note
N3              CON     N1/3                    ' third note
N4              CON     N1/4                    ' quarter note
N8              CON     N1/8                    ' eighth note


' -----[ Variables ]------------------------------------------------------

idx             VAR     Word                    ' loop counter
note1           VAR     Word                    ' first tone for FREQOUT
note2           VAR     Word                    ' second tone for FREQOUT
dur             VAR     Word                    ' duration for FREQOUT
oct1            VAR     Nib                     ' octave for freq1 (1 - 8)
oct2            VAR     Nib                     ' octave for freq2 (1 - 8)
eePtr           VAR     Byte                    ' EEPROM pointer
digit           VAR     Byte                    ' DTMF digit
clkDly          VAR     Word                    ' delay betweens "clicks"


' -----[ EEPROM Data ]-----------------------------------------------------

Phone1          DATA    "916-555-1212", 0       ' a stored telephone number
Phone2          DATA    "916-624-8333", 0       ' another number


' -----[ Initialization ]--------------------------------------------------

Reset:


' -----[ Program Code ]----------------------------------------------------

Main:
  DEBUG CLS,
        "------------------", CR,
        "Sound Effects Demo", CR,
        "------------------", CR, CR
```

```
Dial_Tone:
  DEBUG "Dial tone", CR
  FREQOUT Spkr, 35 */ TmAdj, 35 */ FrAdj        ' "click"
  PAUSE 100
  FREQOUT Spkr, 2000 */ TmAdj,
          350 */ FrAdj, 440 */ FrAdj            ' combine 350 Hz & 440 Hz

Dial_Phone1:                                    ' dial phone from EE
  DEBUG "Dialing number: "
  eePtr = Phone1                                ' initialize eePtr pointer
  GOSUB Dial_Phone

Phone_Busy:
  PAUSE 1000
  DEBUG CR, " - busy...", CR
  FOR idx = 1 TO 6
    FREQOUT Spkr, 400 */ TmAdj,
            480 */ FrAdj, 620 */ FrAdj          ' play 480 Hz and 620 Hz
    PAUSE 620
  NEXT
  FREQOUT Spkr, 35 */ TmAdj, 35 */ FrAdj        ' "click"

Fast_Busy:
  DEBUG " - fast busy...",CR
  FOR idx = 1 TO 8
    FREQOUT Spkr, 200 */ TmAdj,
            480 */ FrAdj, 620 */ FrAdj          ' play 480 Hz and 620 Hz
    PAUSE 310
  NEXT

Dial_Phone2:
  DEBUG "Calling Parallax: "
  eePtr = Phone2
  GOSUB Dial_Phone

Phone_Rings:
  PAUSE 1000
  DEBUG CR, " - ringing"
  PAUSE 2000
  FOR idx = 1 TO 2
    FREQOUT Spkr, 2000 */ TmAdj,
            440 */ FrAdj, 480 */ FrAdj          ' play 440 Hz and 480 Hz
    PAUSE 2000
  NEXT

Camptown_Song:
  DEBUG CR, "Play a Camptown song", CR
  FOR idx = 0 TO 13
    LOOKUP idx, [ G, G, E, G, A, G, E, R, E, D, R, E, D, R], note1
    LOOKUP idx, [ 4, 4, 4, 4, 4, 4, 4, 1, 4, 4, 1, 4, 4, 1], oct1
    LOOKUP idx, [N2,N2,N2,N2,N2,N2,N2,N2,N2,N1,N2,N2,N1,N8], dur
    GOSUB Play_1_Note
  NEXT

Howler:
  DEBUG "Howler -- watch out!!!",CR
```

```
  FOR idx = 1 TO 4
    FREQOUT Spkr, 1000 */ TmAdj,
            1400 */ FrAdj, 2060 */ FrAdj        ' play 1400 Hz and 2060 Hz
    FREQOUT Spkr, 1000 */ TmAdj,
            2450 */ FrAdj, 2600 */ FrAdj        ' play 2450 Hz and 2600 Hz
  NEXT

Computer_Beeps:                                 ' neat with randmom LEDs
  DEBUG "50's Sci-Fi Computer", CR
  FOR idx = 1 TO 50                             ' run about 5 seconds
    RANDOM note1                                ' create random note
    note1 = note1 // 2500                       ' don't go too high
    FREQOUT Spkr, 50 */ TmAdj, note1 */ FrAdj   ' play it
    PAUSE 100                                   ' pause between notes
  NEXT

Space_Transporter:
  DEBUG "Space Transporter", CR
  FOR note1 = 5 TO 5000 STEP 5                  ' frequency sweep up
    FREQOUT Spkr, 10 */ TmAdj,
            note1 */ FrAdj,
            note1 */ 323 */ FrAdj               ' play note, note * 1.26
  NEXT
  FOR note1 = 5000 TO 5 STEP 50                 ' frequency sweep down
    FREQOUT Spkr, 10 */ TmAdj, note1 */ FrAdj
  NEXT
  PAUSE 500

Astro_Droid:                                    ' sounds like R2D2
  DEBUG "Astro Droid", CR
  GOSUB RoboSFX_2                               ' droid squak 1
  GOSUB RoboSFX_3
  GOSUB RoboSFX_6
  PAUSE 1000
  GOSUB RoboSFX_1                               ' droid squak 2
  GOSUB RoboSFX_4
  GOSUB RoboSFX_0
  PAUSE 1000
  GOSUB RoboSFX_3                               ' droid squak 3
  GOSUB RoboSFX_18
  GOSUB RoboSFX_7
  PAUSE 1000
  GOSUB RoboSFX_9                               ' droid squak 4
  GOSUB RoboSFX_12
  GOSUB RoboSFX_10
  PAUSE 1000

Demo_Done:
  DEBUG CR, "End of Sound Effects Demo"
  INPUT Spkr

  END


' -----[ Subroutines ]-----------------------------------------------------

Dial_Phone:
```

```
  DO
    READ eePtr, digit                           ' read a digit
    eePtr = eePtr + 1                           ' update eePtr pointer
    IF (digit = 0) THEN EXIT                    ' when zero, number is done
    DEBUG digit
    IF (digit >= "0") AND (digit <= "9") THEN
      DTMFOUT Spkr, 150 */ TmAdj, 75, [digit - 48]
    ENDIF
  LOOP
  RETURN


Play_1_Note:
  note1 = note1 << (oct1 - 1)                   ' get frequency + octave
  FREQOUT Spkr, dur */ TmAdj, note1 */ FrAdj    ' play it
  RETURN


Play_2_Notes:
  note1 = note1 << (oct1 - 1)                   ' get frequency + octave
  note2 = note2 << (oct2 - 1)                   ' get frequency + octave
  FREQOUT Spkr, dur */ TmAdj,
          note1 */ FrAdj, note2 */ FrAdj        ' play both
  RETURN


' --- Robot Sound Effects ---

RoboSFX_0:
  FOR note1 = 1 TO 4
    FOR note2 = 2000 TO 50 STEP 400
      FREQOUT Spkr, 10 */ TmAdj, note2 */ FrAdj
    NEXT
    FOR note2 = 800 TO 2000 STEP 400
      FREQOUT Spkr, 10 */ TmAdj, note2 */ FrAdj
    NEXT
  NEXT
  RETURN


RoboSFX_1:
  FOR note1 = 800 TO 2000 STEP 100
    FREQOUT Spkr, 10 */ TmAdj, note1 */ FrAdj
  NEXT
  FOR note1 = 2000 TO 50 STEP 100
    FREQOUT Spkr, 10 */ TmAdj, note1 */ FrAdj
  NEXT
  RETURN


RoboSFX_2:
  FOR note1 = 1000 TO 40 STEP 20
    FREQOUT Spkr, 10 */ TmAdj, note1 */ FrAdj
  NEXT
  RETURN
```

```
RoboSFX_3:
  FOR note1 = 10000 TO 500 STEP 500
    FREQOUT Spkr, 10 */ TmAdj, note1 */ FrAdj
  NEXT
  RETURN


RoboSFX_4:
  FOR note1 = 10 TO 50 STEP 10
    FOR note2 = 50 TO 10 STEP 50
      FREQOUT Spkr, 15 */ TmAdj, (note2 * 20) */ FrAdj
    NEXT
  NEXT
  RETURN


RoboSFX_5:
  FOR note1 = 1 TO 120 STEP 2
    FREQOUT Spkr, 10 */ TmAdj, (SIN(note1 + 40) * 50) */ FrAdj
  NEXT
  RETURN


RoboSFX_6:
  FOR note1 = 10 TO 50 STEP 10
    FOR note2 = 50 TO 10 STEP 50
      FREQOUT Spkr, 10 */ TmAdj, (note1 * note2) */ FrAdj
    NEXT
  NEXT
  RETURN


RoboSFX_7:
  FOR note1 = 30 TO 70 STEP 5
    FOR note2 = 70 TO 30 STEP 5
      FREQOUT Spkr, 10 */ TmAdj, (note1 * note2) */ FrAdj
    NEXT
  NEXT
  RETURN


RoboSFX_8:
  FOR note1 = 30 TO 60 STEP 10
    FOR note2 = 60 TO 30 STEP 10
      FREQOUT Spkr, 10 */ TmAdj, (note1 * note2) */ FrAdj
    NEXT
  NEXT
  RETURN


RoboSFX_9:
  FOR note1 = 1 TO 60 STEP 7
    FREQOUT Spkr, 10 */ TmAdj, (SIN(note1 + 20) * 30) */ FrAdj
  NEXT
  RETURN
```

```
RoboSFX_10:
  FOR note1 = 1 TO 30
    FREQOUT Spkr, 20 */ TmAdj, ((note1 * 14) + 450) */ FrAdj
  NEXT
  RETURN


RoboSFX_11:
  FOR note1 = 10000 TO 500 STEP 500
    FREQOUT Spkr, 20 */ TmAdj, note1 */ FrAdj
  NEXT
  RETURN


RoboSFX_12:
  FOR note1 = 102 TO 82 STEP 2
    FREQOUT Spkr, 40 */ TmAdj,
            ((COS(note1 / 100) + 36) * 25) */ FrAdj
    FREQOUT Spkr, 20 */ TmAdj,
            ((SIN(note1 / 100) + 20) * 50) */ FrAdj
  NEXT
  RETURN


RoboSFX_13:
  FOR note1 = 1 TO 10
    FREQOUT Spkr, 40 */ TmAdj, 1195 */ FrAdj
    FREQOUT Spkr, 40 */ TmAdj, 2571 */ FrAdj
  NEXT
  RETURN


RoboSFX_14:
  FOR note1 = 1 TO 3
    FREQOUT Spkr, 90 */ TmAdj, 550 */ FrAdj
    FREQOUT Spkr, 90 */ TmAdj, 400 */ FrAdj
  NEXT
  RETURN


RoboSFX_15:
  FOR note1 = 40 TO 15
    FREQOUT Spkr, 5 */ TmAdj, (note1 * 90) */ FrAdj
    FREQOUT Spkr, 5 */ TmAdj, (note1 * 80) */ FrAdj
    FREQOUT Spkr, 5 */ TmAdj, (note1 * 70) */ FrAdj
    FREQOUT Spkr, 5 */ TmAdj, (note1 * 60) */ FrAdj
    FREQOUT Spkr, 5 */ TmAdj, (note1 * 50) */ FrAdj
  NEXT
  RETURN


RoboSFX_16:
  FOR note1 = 1 TO 20
    FREQOUT Spkr, 20 */ TmAdj,
            (1195 - (note1 * 50)) */ FrAdj
    FREQOUT Spkr, 20 */ TmAdj,
            (1195 + (note1 * 50)) */ FrAdj
```

```
  NEXT
  RETURN


RoboSFX_17:
  FOR note1 = 0 TO 150 STEP 10
    FREQOUT Spkr, 20 */ TmAdj, (1295 - note1) */ FrAdj
    FREQOUT Spkr, 20 */ TmAdj, (1095 + note1) */ FrAdj
  NEXT
  RETURN


RoboSFX_18:
  FOR note1 = 1 TO 20
    FREQOUT Spkr, 10 */ TmAdj, (note1 * 50)  */ FrAdj
    FREQOUT Spkr, 10 */ TmAdj, (note1 * 100) */ FrAdj
    FREQOUT Spkr, 10 */ TmAdj, (note1 * 150) */ FrAdj
  NEXT
  RETURN
```

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Parallax:
29143