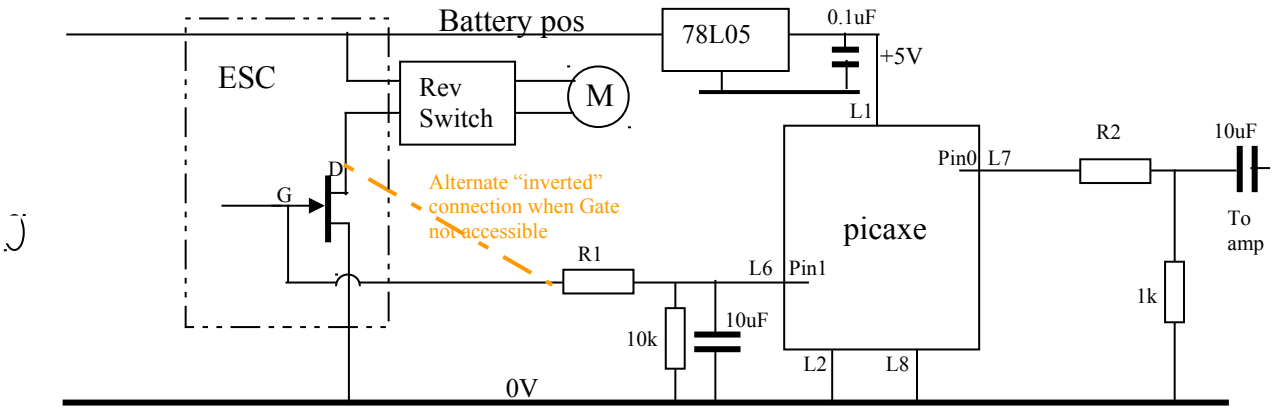
The SANDSTONE & TERMITE RAILWAY
"Linking the Outback to the World"

PICAXE TUTORIALS

2. Sound for Battery operated Steam Locos

NOTE: All these programs are for battery operated locos. It <u>should</u> be possible to use them on track power by using a battery to supply the amplifier and Picaxe, and a bridge rectifier from the motor terminals to the input R1 in the diagram. I have not checked this aspect.

Simplest sound – no volume control and the same cct for both diesel and steam



The sound appears at L7 ('pin0') and is fed into a small amplifier - LM386 in this case, but any could be used.
(I found that the diesel sound needs a higher gain for the same perceived volume, probably because it is a lower frequency. So you could use an LM380 for the diesel.)

The basic program flow is:-
First define variables:
SYMBOL speed=w1 (put the value of speed into word1 ie bytes2&3)

Read in the voltage that represents the speed from pin1
READADC10 1, speed
Where 'speed' is a memory variable which will hold the voltage in 'counts' from 0 to 1023 for 0 to 5V in.
I find it easier to use a byte for numbers where possible, and I like to use say 100 as a nice round number for speed. So we can reduce the speed value to a max 100, by for example: Speed=speed/10, but for a more exact equation see later:

(note in this case we still are using 2 bytes=word for speed, but we'll need it as a byte later!)
We now have a number approximating the speed of the loco, if we use the 'normal' connection in the circuit (ie FET gate) or,
a number representing the inverse of speed if using the connection to the ESC/FET drain. (speed decreases as motor voltage increases.)
so our program has the form of:

SYMBOL maxspeedIN=255 - see note below for determining value of maxspeedIN

Then to take this into account and ensure we always have 100 (say) for the max speed, use:
speed=speed*25 /maxspeedIN max 100

and if using alternative FET connection, then add this line:
speed=100-speed.
NOTE: if using this alternative connection, as the battery voltage reduces during train running, the chuff rate will become faster than the loco seems to be going. It will get to the point that it never stops chuffing even when the loco is stopped! This is a good surrogate low voltage alarm!
This is because we have assumed a constant maxspeedIn value. See *Advanced Picaxe Motor Control.doc* on how to overcome this.

Warning:
the max volts applied to pin1 for the speed MUST NOT EXCEED the picaxe supply voltage assuming that is 5V, then a voltage divider on pin1 must reduce the motor volts to less than or equal to 5V at the MAXIMUM SUPPLY VOLTAGE ! It would be prudent to reduce the voltage to a bit less than 5V

Determining maxspeedIN
If max volts on pin1 are less than picaxe volts (5V) then you must determine the value ***for maxspeedIN*** *as follows:*
*using a multimeter measure the picaxe supply volts (Vp) and the voltage on pin1 with max supply voltage (Vm1). Then maxspeedIN= Vm1/Vp*256. (round the result UP, and must be max of 256.)*
example, you measure Vp=4.28V and Vm1=2.65V, then enter SYMBOL maxspeedIN=159

Steam Chuff Program:
We will just produce a white noise chuff for a period (I'll call it chufftime) and then silence (offtime) for the same period and repeat. The period will depend on how fast the loco is going.
Calculate the length of the chuff (white noise) and the length of the 'offtime' between chuffs. (it's easiest to make these equal to start with) from the speed. So define:
SYMBOL chufftime=b4
SYMBOL offtime=w3 this needs to be a Word as can be>255

For steam chuff, use the SOUND command with a note of 255 (white noise), but any number from 128 to 255 produces a 'sort of' white noise.

SOUND pin, (255, chufftime). The duration is in 12ms periods (at clock of 4 MHz).
Because the SOUND command uses 12 ms 'periods', the value of offtime will be 12 times that of chufftime.

Now the length of offtime and chufftime will be inversely proportional to the speed (the faster you go the shorter the chuff),so we can use something like this:

offtime=speedconstant/speed (in ms) and
chufftime=offtime/12

where the constant has to be determined to suit the loco gearing, wheels etc.
Note that somewhere we'll have to check we aren't dividing by zero (speed)! So we'll introduce a variable called stoppedcounts and 'define' a speed less than this as being 'stopped'. I usually use a number around 5% of max.
SYMBOL stoppedcounts=5 (if using 100 as max speed)

We'll try letting
SYMBOL speedconstant=3000,
Then at max speed (100) we'll get offtime=30ms and chufftime=2.5 which gives 2
This means the total chuff will take 54ms and there are 4 chuffs per wheel revolution, so one wheel revolution will be 216ms, or 278 RPM. For a 5' dia wheel, this gives a *simulated* 50 mph top speed.
Now at slow speed, so 5% of max, we'll have a speed count reading of 5, so offtime= 600ms and chufftime=50 representing a speed of 2 mph.
(As a rough approximation, and mixing dimensions, **chuff period (sec) = wheel dia in ft / kph)**

You have to choose (determine) the value of 'speedconstant' to make the chuffs appear to occur 4 times for every revolution of the driving wheels. This will not be possible at all speeds because the speed is not exactly represented by the voltage we read in. I find it is best if you choose constant to 'look right' at slow speeds.

Then we then go back to the start and repeat.

NOTE: the SOUND command just produces a sequence of pulses of magnitude +5V, with the on and off times varying randomly.

You can make a reasonable steam whistle using SOUND pin,(117,100) (but there will be no chuff while it is sounding!) Try experimenting with the value of 117 - 110 to 125 might be OK.

Here's a full prog listing:

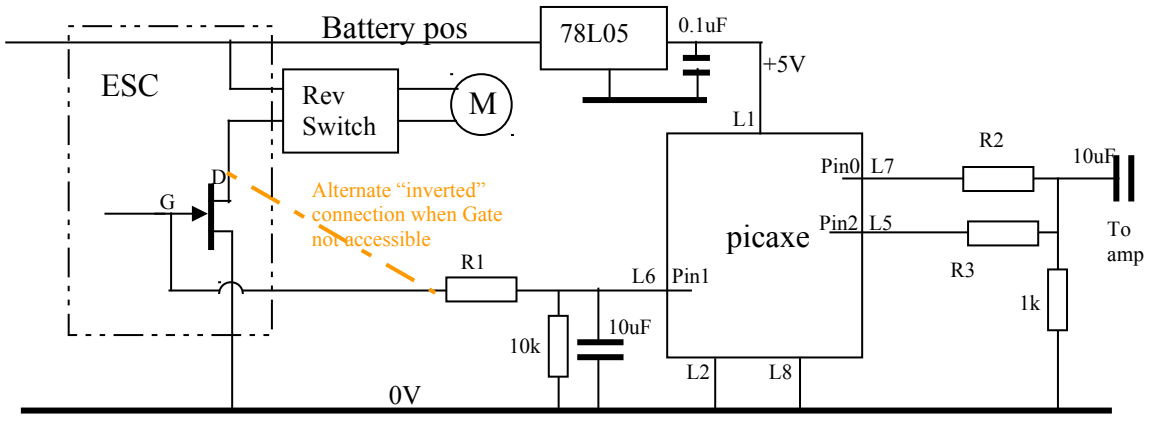
```
 Simplestchuff.bas
'example of the simplest steam sound generator
'Greg Hunter    6/4/14
'a voltage on pin1 represents the speed of the loco.
'a 'white noise' chuff sound is produced on pin0, which must be amplified externally.
'suitable for 08M  and 08M2   at 4 MHz


'pin0 is sound output
'pin1 is speed voltage input    0V=stopped, +5V=max speed
                               'if speed volts are inverted ie 5V=stopped and 0V=max speed, then
                               'invoke line INVspd:


'define variables
        'w0=b0,b1 not used
        SYMBOL speed=w1             'will be 0-maxspeedcounts in prog after reading in and
                                        processing
        SYMBOL chufftime=b4         'length of chuff in 12 ms periods (use in SOUND command)
        'b5 not used
        SYMBOL offtime=w3                   'in ms. need a word as can be >255 at slow speed

'define constants
        SYMBOL maxspeedcounts=100       'max speed will be represented as this in prog.
        SYMBOL stoppedcounts=5          'define less than this as 'stopped'= about 5% of
maxspeedcounts
        SYMBOL maxspeedIN=255           'see notes at end to calc this value
        SYMBOL speedconstant=3000       'adjust for 4 chuffs per wheel revolution
                                        'depends on maxspeedcounts too.
```

```
'-------------------------------------------------------------------------------------------------------

start:
        READADC10 1, speed
'now adjust for actual input voltage and normalise to a specified maximum (maxspeedcounts)
        speed=speed/4*maxspeedcounts/maxspeedIN MAX maxspeedcounts

'INVspd:    speed=maxspeedcounts-speed      'only use if speed voltage is inverted

'from here in prog, speed is 0 to maxspeedcounts

        if speed<stoppedcounts then stopped:          'define as stopped - do constant hiss
        offtime=speedconstant/speed  '(in ms) and
        chufftime=offtime/12            'in 12ms counts

        SOUND 0, (255, chufftime)
        PAUSE offtime
        goto start

stopped:
        SOUND 0, (255, 100)
        goto start
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Volume control

It would be nice to control the volume, so that for instance, the chuffs are quieter when not accelerating, or there is a soft hiss when stopped.
A simple way would be to control a relay and use its contacts to short out a resistor in series with the speaker. But that's not very elegant – I'd prefer to use the Picaxe directly.
I've used a couple of circuits to do this:

The simplest idea is to send the SOUND output to two different pins which connect to circuits that provide different gains.



Try using 10k for R2 and 47k for R3. I've found that a quiet level of about ¼ of the loud is about right.

When you want loud you use

SOUND 0,(255,chufftime)

When you want softer volume use
SOUND 2, (255,chufftime)

Now how to get the program to decide if we want loud or soft?
I decided I wanted the volume to be soft when speed was decreasing and when loco is stopped.
So I use two new variables called 'oldspeed1' and 'oldspeed2' to store the speed from the two previous readings. If the 'oldspeeds' are less than the latest 'speed', then we want it loud. Otherwise we want soft. Because speed is in the range 0-100, oldspeed only needs to be a byte, so define:-
SYMBOL oldspeed1=b10
SYMBOL oldspeed2=b11

I found it necessary to use two oldspeeds because you often get a 'jitter' of +/-1 in the speed reading.
Then define another variable to hold the pin number we want the sound on (0 or 2).
SYMBOL soundpin=b8

Then the basic program flow is like this:
oldspeed2=oldspeed1
oldspeed1=speed
Then use **READADC10** 1, speed etc as before to get speed 0-100

Now conditions for loud are
1) just starting from stopped, or
2) speed is increasing

IF speed<stoppedcounts **THEN GOTO** stopped
IF oldspeed1=0 and speed>stoppedcounts **THEN GOTO** loud
IF speed>=oldspeed1 AND oldspeed1>oldspeed2 **THEN GOTO** loud
IF speed<oldspeed1 AND oldspeed1<oldspeed2 **THEN GOTO** soft
Goto rest of program (if we get here, we just want to leave volume as it was)

Soft: Soundpin=2
Goto rest of program

Loud: Soundpin=0
Goto rest of program

And the rest of program will be similar to before except the sound command will look like:
SOUND soundpin, (255,chufftime) etc

In the diesel sound program, the HIGH and LOW commands will become:
HIGH soundpin and **LOW** soundpin

Sounds when stopped

I like to have a soft hiss sound when a steam loco is stopped to remind me the loco is still turned on.
So I usually defined 'stopped' as being less than 5% of max voltage read in.
So we put in a line like this:
IF speed<stoppedcounts **THEN GOTO** stopped: (goto the subroutine called 'stopped')

For steam sound:
Stopped:
soundpin=2
SOUND soundpin, (255,50) which will play white noise for 600ms before it goes and reads the input speed again.
GOTO start:

Listing for:-
SimpleVolumeChuff.bas

```
'example of the simple steam sound generator WITH VOLUME CONTROL AND STOPPED HISS
'Greg Hunter    8/4/14
'a voltage on pin1 represents the speed of the loco.
'a 'white noise' chuff sound is produced on pin0 and pin2, which must be amplified externally.
'suitable for 08M  and 08M2   at 4 MHz


'pin0 is LOUD sound output
'pin1 is speed voltage input    0V=stopped, +5V=max speed
                                'if speed volts are inverted ie 5V=stopped and 0V=max speed, then
                                'invoke line INVspd:
'pin2 is SOFT sound output


'define variables
        'w0=b0,b1 not used
        SYMBOL speed=w1         'will be 0-maxspeedcounts in prog after reading in and processing
        SYMBOL chufftime=b4     'length of chuff in 12 ms periods (use in SOUND command)
        '=b5
        SYMBOL offtime=w3               'in ms. need a word as can be >255 at slow speed
        SYMBOL soundpin=b8              'pin0 for loud, pin2 for soft
        SYMBOL oldspeed1=b10
        SYMBOL oldspeed2=b11
'define constants
        SYMBOL maxspeedcounts=100       'max speed will be represented as this in prog.
        SYMBOL stoppedcounts=5          'define less than this as 'stopped'= about 5% of
maxspeedcounts
        SYMBOL maxspeedIN=255           'see notes at end to calc this value
        SYMBOL speedconstant=3000       'adjust for 4 chuffs per wheel revolution
                                        'depends on maxspeedcounts too.
'-------------------------------------------------------------------------------------------------


start:
          oldspeed2=oldspeed1
          oldspeed1=speed

        READADC10 1, speed
'now adjust for actual input voltage and normalise to a specified maximum (maxspeedcounts)
        speed=speed/4*maxspeedcounts/maxspeedIN MAX maxspeedcounts


'INVspd:   speed=maxspeedcounts-speed     'only use if speed voltage is inverted
```

```
'*now speed is always represented by 0 when stopped and maxspeedcounts at max input voltage*

          if speed<stoppedcounts then stopped
L20:      if OLDspeed1=0 and speed>stoppedcounts then loud
L35:      if speed>=OLDspeed1 AND OLDspeed1>OLDspeed2 then loud
L40:      if speed<OLDspeed1 AND OLDspeed1<OLDspeed2 then soft
'otherwise just leave as it was


chuff:
        offtime=speedconstant/speed  '(in ms) and
        chufftime=offtime/12         'in 12ms counts

        SOUND soundpin, (255, chufftime)
        PAUSE offtime

        goto start

stopped:
        soundpin=2
        speed=0
        oldspeed1=0
        SOUND soundpin, (255, 75)  'constant hiss
        goto start

loud:
        soundpin=0
        goto chuff
soft:
        soundpin=2
        goto chuff
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'NOTES:
'the max volts applied to pin1 for the speed MUST NOT EXCEED the picaxe supply voltage.
'Assuming that is 5V, then a voltage divider on pin1 must reduce the motor volts to
'less than or equal to 5V at the MAXIMUM SUPPLY VOLTAGE !
'It would be prudent to reduce the voltage to a bit less than 5V

'if max volts on pin1 are less than picaxe volts (5V) then you must determine the value
'for maxspeedIN as follows:
'using a multimeter measure the picaxe supply volts (Vp) and
'the voltage on pin1 with max supply voltage (Vm1). Then
'maxspeedIN= Vm1/Vp*256. (round the result UP, and must max of 256.)
'example,  you measure Vp=4.28V and Vm1=2.65V, then enter SYMBOL maxspeedIN=159
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Chuffs with a 'beat'

Maybe you'd like the chuffs to have slightly different lengths to simulate valve gear that's 'not quite right'.
To make every 4th chuff offtime a bit longer, try adding this:

SYMBOL chuffcounter=b5

start: chuffcounter=chuffcounter+64 (values of chuffcounter are 0,64,128,192, then 256=0 again)

then in the part of the program where you calculate offtime, add
offtime=speedconstant/speed (normal calc)
if chuffcounter<>0 then goto nextline (for 3 chuffs use normal offtime)
offtime=offtime*5/4 (4^{th} chuff, lengthen offtime by 20%)

nextline: this is the continuation of the program

full listing of:-
chuffwithbeat.bas

```
'example of the simple steam sound generator with volume controland stopped hiss,
'and a longer chuff every 4th chuff to give a 'beat'
'Greg Hunter    8/4/14
'a voltage on pin1 represents the speed of the loco.
'a 'white noise' chuff sound is produced on pin0 and pin2, which must be amplified externally.
'suitable for 08M  and 08M2   at 4 MHz


'pin0 is LOUD sound output
'pin1 is speed voltage input.   0V=stopped, +5V=max speed
                               'if speed volts are inverted ie 5V=stopped and 0V=max speed, then
                               'invoke line INVspd:
'pin2 is SOFT sound output


'define variables
        'w0=b0,b1 not used
        SYMBOL speed=w1          'will be 0-maxspeedcounts in prog aftere reading in and
processing
        SYMBOL chufftime=b4      'length of chuff in 12 ms periods (use in SOUND command)
        SYMBOL chuffcounter=b5   'counts 1 to 4 chuffs
        SYMBOL offtime=w3        'in ms. need a word as can be >255 at slow speed
        SYMBOL soundpin=b8       'pin0 for loud, pin2 for soft
        'SYMBOL delta=b9         'the change in speed between readings
        SYMBOL oldspeed1=b10
        SYMBOL oldspeed2=b11
'define constants
        SYMBOL maxspeedcounts=100      'max speed will be represented as this in prog.
        SYMBOL stoppedcounts=5         'define less than this as 'stopped'= about 5% of
maxspeedcounts
        SYMBOL maxspeedIN=255          'see notes at end to calc this value
        SYMBOL speedconstant=3000      'adjust for 4 chuffs per wheel revolution
                                       'depends on maxspeedcounts too.
'-------------------------------------------------------------------------------------------------

start:
          chuffcounter=chuffcounter+64
          oldspeed2=oldspeed1
          oldspeed1=speed
```

```
      READADC10 1, speed
'now adjust for actual input voltage and normalise to a specified maximum (maxspeedcounts)
        speed=speed/4*maxspeedcounts/maxspeedIN MAX maxspeedcounts


'INVspd:   speed=maxspeedcounts-speed      'only use if speed voltage is inverted

'*now speed is always represented by 0 when stopped and maxspeedcounts at max input voltage*

          if speed<stoppedcounts then stopped
L20:      if OLDspeed1=0 and speed>stoppedcounts then loud
L35:      if speed>=OLDspeed1 AND OLDspeed1>OLDspeed2 then loud
L40:      if speed<OLDspeed1 AND OLDspeed1<OLDspeed2 then soft
'(could try having speed<=oldspeed1 in the above line)

'otherwise just leave volume as it was


chuff:
        offtime=speedconstant/speed  '(in ms) and
        chufftime=offtime/12            'in 12ms counts
'now give a 'beat'
        if chuffcounter<>0 then CH2:
        offtime=offtime*5/4   'lengthen offtime on 4th beat

CH2:   SOUND soundpin, (255, chufftime)
        PAUSE offtime

        goto start

stopped:
        soundpin=2
        speed=0
        oldspeed1=0
        SOUND soundpin, (255, 75)  'constant hiss
        goto start

loud:
        soundpin=0
        goto chuff
soft:
        soundpin=2
        goto chuff
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'NOTES:
'the max volts applied to pin1 for the speed MUST NOT EXCEED the picaxe supply voltage.
'Assuming that is 5V, then a voltage divider on pin1 must reduce the motor volts to
'less than or equal to 5V at the MAXIMUM SUPPLY VOLTAGE !
'It would be prudent to reduce the voltage to a bit less than 5V

'if max volts on pin1 are less than picaxe volts (5V) then you must determine the value
'for maxspeedIN as follows:
```

'using a multimeter measure the picaxe supply volts (Vp) and
'the voltage on pin1 with max supply voltage (Vm1). Then
'maxspeedIN= Vm1/Vp*256. (round the result UP, and must max of 256.)
'example, you measure Vp=4.28V and Vm1=2.65V, then enter SYMBOL maxspeedIN=159

Varying the valve 'cutoff'

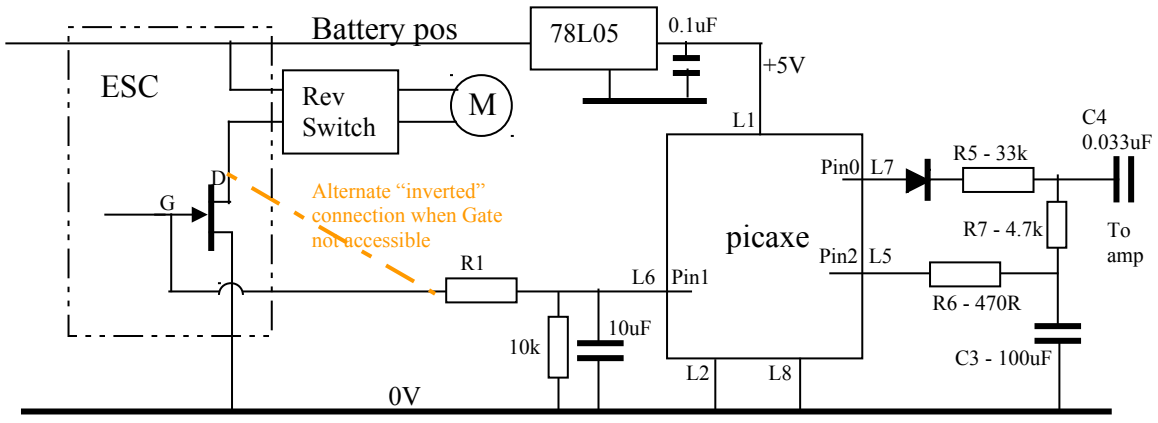
So far, we have assumed the chufftime and offtime are equal. A real loco admits steam to cylinders for longer at starting and for less time at higher speed.
This will have the chuffs about 75% of total time at starting and about 25% at max speed
b9=speedconstant/maxspeedcounts (b9 is a temporary, intermediate value)
offtime=speedconstant/2/speed+b9
chufftime=2*speedconstant/speed-offtime/12

Using speedconstant=3000, this will give a chufftime of 72 counts at 5% speed and 1 count at max speed (100), and offtime of 330ms at starting and 45ms at top speed

But I find the effect is not very noticeable.
(no program written)

Shaping the steam chuff

So far the chuffs we produce are simple on and off and sound quite 'harsh'. What we need is a 'shaper' to give an slow build up and turn off for the chuff – attack and decay time. This means we need to vary the volume in a continuous manner, not just loud and soft.
An external circuit to do this is here:



In this case, we output a varying (PWM) voltage on pin2 to control the volume, which is filtered to produce DC.
The white noise chuff is always output on pin0.
R6 and C3 filter the PWM signal representing the volume required, so that the voltage on C3 varies from 0 to 5V. The diode and R5 and R7 form a voltage divider of about 1/6.

The voltage on C3 is subtracted from the 5V peak of the white noise to give a white noise of amplitude (5-V_{C5}). So 0V on pin2 is Max volume and 5V on pin2 is min volume.
The dc level is removed by C4 and the signal amplified by 20 in the LM386.

The timeconstant of R6C3 provides a ramp up and ramp down of the volume for the attack and decay time of the chuff. I have used 47 ms. R6 is chosen to limit the max current from the Picaxe, then R7 is ten times that.
The program now has to

- turn on the chuff
- Increase the volume by decreasing the PWM voltage on pin2
- Before the 'end' of the chuff, decrease the volume by increasing PWM on pin2
- End the chuff

The previous loud and soft chuff at accelerate/slowing can also be achieved using the PWM on pin2. so the program flow for each chuff looks like:

```
SYMBOL volume=w6                (0=loud, 255=min vol, but needs to be a Word)

PWMOUT 2,63, loudVol            'at 15.7kHz,  255=min vol, 0=max vol
SOUND 0, (255, chufftime)

'now turn off the volume and let it decay with ext cct
PWMOUT 2,63,offvol
SOUND 0, (255, offtime)      'sounds much better than having silence during offtime
```

(note: offtime is now in 12ms sound periods)

But in this case the calculations for chufftime and offtime may need to be slightly different from previous, as they include the attack and decay time.
I chose the the PWMOUT frequency to be 15.7 kHz to be above hearing and to give a max volume number of 255.
We also need to define

```
SYMBOL loudVol=150       (need to experiment with these values)
SYMBOL offVol=250
```

full listing - (has not been tested in a circuit yet!!!)

```
CHUFFshaping.bas
'example of the steam sound generator WITH VOLUME CONTROL and SHAPING
'Greg Hunter    9/4/14          100 bytes
'**not tested in actual circuit ******

'a voltage on pin1 represents the speed of the loco.
'a 'white noise' chuff sound is produced on pin0 and pin2, which must be amplified externally.
'suitable for 08M  and 08M2   at 4 MHz


'pin0 is sound output
'pin1 is speed voltage input    0V=stopped, +5V=max speed
                                'if speed volts are inverted ie 5V=stopped and 0V=max speed, then
                                'invoke line INVspd:
'pin2 is PWM output voltage to control volume and shape the chuff
```

```
'define variables
        'w0=b0,b1 not used
        SYMBOL speed=w1          'will be 0-maxspeedcounts in prog after reading in and processing
        SYMBOL chufftime=b4      'length of chuff in 12 ms periods (use in SOUND command)
        '=b5
        SYMBOL offtime=w3                'in ms. need a word as can be >255 at slow speed
        SYMBOL oldspeed1=b10
        SYMBOL oldspeed2=b11
        SYMBOL PWMvolume=w6     '0=max volume, 255 =min volume
                                 'has to be a word, but using 15kHz PWM, max value is 255
'define constants
        SYMBOL maxspeedcounts=100       'max speed will be represented as this in prog.
        SYMBOL stoppedcounts=5          'define less than this as 'stopped'= about 5% of
maxspeedcounts
        SYMBOL maxspeedIN=255           'see notes at end to calc this value
        SYMBOL speedconstant=3000       'adjust for 4 chuffs per wheel revolution
                                        'depends on maxspeedcounts too.
        'determine these by experiment.....
        SYMBOL loudVol=150              'chuff volume when accelerating
        SYMBOL offVol=250               'hiss volume when stopped and between chuffs
        SYMBOL midVol=210               'chuff volume when slowing
'-----------------------------------------------------------------

start:
          oldspeed2=oldspeed1
          oldspeed1=speed

        READADC10 1, speed
'now adjust for actual input voltage and normalise to a specified maximum (maxspeedcounts)
        speed=speed/4*maxspeedcounts/maxspeedIN MAX maxspeedcounts


'INVspd:   speed=maxspeedcounts-speed     'only use if speed voltage is inverted

'*now speed is always represented by 0 when stopped and maxspeedcounts at max input voltage*

          if speed<stoppedcounts then stopped
L20:      if OLDspeed1=0 and speed>stoppedcounts then loud
L35:      if speed>=OLDspeed1 AND OLDspeed1>OLDspeed2 then loud
L40:      if speed<OLDspeed1 AND OLDspeed1<OLDspeed2 then soft
'otherwise just leave as it was


chuff:
        offtime=speedconstant/speed  '(in ms) and
        chufftime=offtime/12         'in 12ms counts
        offtime=offtime/12           'needs to be in 12ms counts now.
'start chuff shape charging capacitor
        PWMOUT 2,63, PWMvolume                  'at 15.7kHz,  255=min vol, 0=max vol

        SOUND 0, (255, chufftime)   'start hiss sound
```

```
        PWMOUT 2,63,offvol              'start to decay chuff
        SOUND 0, (255, offtime)
        goto start

stopped:
        speed=0
        oldspeed1=0
        PWMOUT 2,63,offvol              'soft volume
        sound 0, (255, 100)             '3/4 sec of hiss
        goto start

loud:   PWMvolume=loudVol
        goto chuff
soft:   PWMvolume=midVol
        goto chuff
'+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

'NOTES:
'the max volts applied to pin1 for the speed MUST NOT EXCEED the picaxe supply voltage.
'Assuming that is 5V, then a voltage divider on pin1 must reduce the motor volts to
'less than or equal to 5V at the MAXIMUM SUPPLY VOLTAGE !
'It would be prudent to reduce the voltage to a bit less than 5V

'if max volts on pin1 are less than picaxe volts (5V) then you must determine the value
'for maxspeedIN as follows:
'using a multimeter measure the picaxe supply volts (Vp) and
'the voltage on pin1 with max supply voltage (Vm1). Then
'maxspeedIN= Vm1/Vp*256. (round the result UP, and must max of 256.)
'example,  you measure Vp=4.28V and Vm1=2.65V, then enter SYMBOL maxspeedIN=159
```