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

PICAXE TUTORIALS

1. Introduction to using Picaxe for model Train control and sound.

Introduction to Picaxe

Picaxe are microcontrollers that you program using a desktop computer. They are super useful for controlling model trains and making sounds, with few external components.

To save my repeating it, see the website for an introduction:
http://www.picaxe.com/What-Is-PICAXE

To get started you're going to need
A computer (desktop/laptop)
A free downloaded Program Editor to allow you to write Picaxe programs in a form of BASIC
A USB or COM cable, to connect to…..
A 'development board' for downloading the program to the Picaxe, debugging your program and testing your external interface circuit.

Warning: If you don't feel comfortable with programming and building simple external electronic circuits, this is not for you!

Once you've downloaded the Programming Editor, go to Help and see what's available. Manual 2 – BASIC commands will list all the commands.

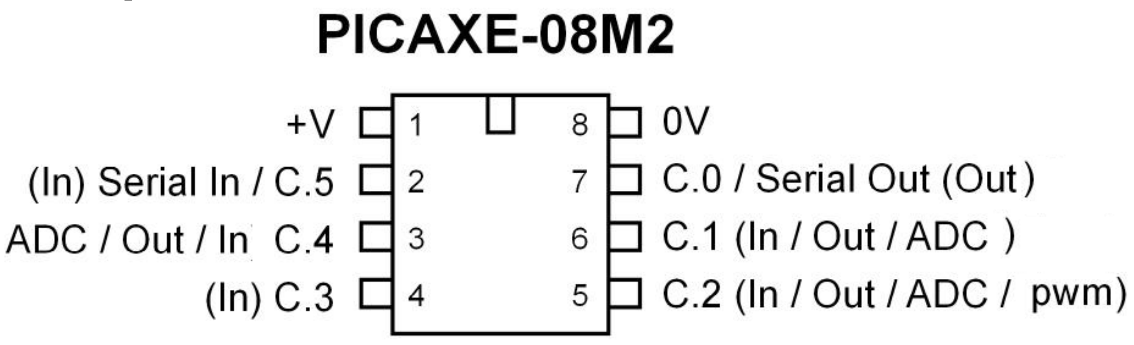
Why it is useful for model trains?

The 08xx Picaxe has 5 inputs and outputs, and in the program, you can configure these to be:-
1 in, 4 out, or
2 in, 3 out, or
3 in, 2 out, or
4 in, 1 out.
In the simplest form these are just digital inputs – you put either 0V or +5V on them and read the state with the program.
Some pins have an analogue to digital converter (ADC) so you can read actual external voltages into the program.
There is a **PWMOUT** command to control a motor's speed. This is a great command as it runs 'in the background' - you can have the program doing other things while the motor command just goes on.
There is a similar **SERVO** command to control a servo motor
There is a **SOUND** command that can produce 128 frequencies (useful for horns) and white noise – just what the doctor ordered for steam chuffs.
There are the usual commands like **IF..THEN** and **FOR…TO** to allow you to test conditions and do things multiple times.

And that's about all we need to control a train or make noise.

All the following comments and programs are aimed at battery operated trains, not track power.

Pin labelling
The one giant mistake that the Picaxe developers made is to use the word 'pin' within the program, to denote internal program references to external pins of the chip. Usually the word 'pin' denotes the actual physical pin of the chip.
To 'overcome' this, they suggest using the word 'leg' to refer to the actual external pin of the chip
I will use this convention, so that on an 8pin chip (Picaxe 08xx) (external)leg7=(program)pin0 etc.
This is the pinout for the



Note the labeling of pins as C.0, C.1 etc. For the 8 pin picaxes, it is not necessary to use the **C.** prefix - and just using 0 , 1 etc will work OK.

Maths
One other complication is the 'maths' available. It is purely 'integer maths'. ie if you divide 8 by 3 you get 2 because the true answer 2.666 can't be represented. Also the maths is performed strictly in the order it's written – not using the normal rules of 'multiply before add' so if you write

X=3*Y+7, you will get the correct answer (say Y=10, then X=37). But if you write
X=7+3*Y you will get (7+3)*Y = 100

IF..THEN statements
Another quirk of the Basic language used is that in the **IF…THEN** statement, you cannot do anything except GOTO. ie you cannot write IF X=Y THEN X=10; you must use a goto such as:

IF X=Y THEN (goto) label1:
And label1: will say X=10.

<u>I am only going to deal with the 8 pin Picaxe as this is all I have used. More pins just provide more input and output pins.</u>

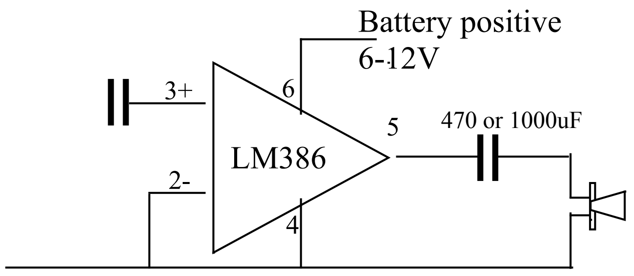
Programming circuits
Picaxe uses leg2 and leg7 for programming, with a couple of resistors on leg2. leg7 (pin0) is available as an output from the Picaxe, but leg2 (pin5) is generally not used except for programming. Picaxe designers always suggest you include the 2 resistors and a socket on leg2 in all your circuits so you can reprogram the Picaxe in-situ. **I don't do this**.
All my circuits have an 8 pin IC socket which the Picaxe plugs into, and all my programming is done on my development board. Leg2 is permanently connected to 0V on every one of my circuits. I then just take the Picaxe from the development board and plug it into the IC socket in my circuit. If I need to change the program, I unplug the Picaxe from its socket and take it to the development board, reprogram it, test it and return it to its circuit.
I will use capitals to denote the commands available in the Picaxe language.

Power supply

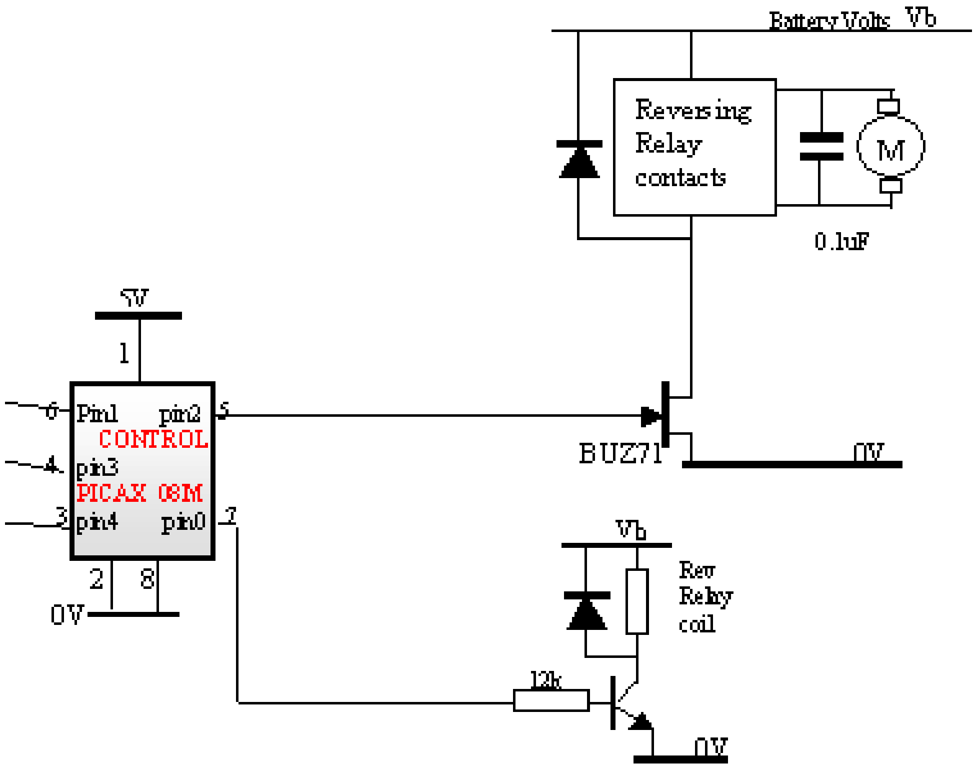
Picaxe can run from 3.3V to 5V. I run all my circuits from 5V, usually using a 78L05 regulator.
I always put a 0.1 uF capacitor across the legs 1 and 8 for spike suppression.
Each pin can supply 20 mA with a total chip load of 80mA.

Amplifiers

As the Picaxe only puts out 5V, you will always need some external circuitry to control a motor or drive a loudspeaker. A typical simple amp uses an LM386 chip for about 1/2W into 8 ohm speaker.



A typical motor control uses a FET rated at say 5 amps. (say BUZ71 or   IRFZ44)
A small reversing relay with a coil rated at 12V can be drive using a small transistor like a BC548.
So a generic type of circuit might look like this:



Byte usage in these example programs

word	byte	Steam chuff	diesel	Motor control
w0	b0		Seed (shift register)	Timecounts (pulsin)
	b1			
w1	b2	Speed input voltage	Speed input voltage	PWMspeed output
	b3			
w2	b4	chufftime	pausetime	
	b5	chuffcounter	Lookuptable ?	
w3	b6	offtime (ms)		
	b7			
w4	b8	soundpin	soundpin	
	b9	Temporary in cutoff calcs		
w5	b10	oldspeed1	oldspeed1	
	b11	oldspeed2	oldspeed2	
w6	b12	PWMvolume (PWM 15kHz)	PWMvolume (PWM 15kHz)	
	b13			