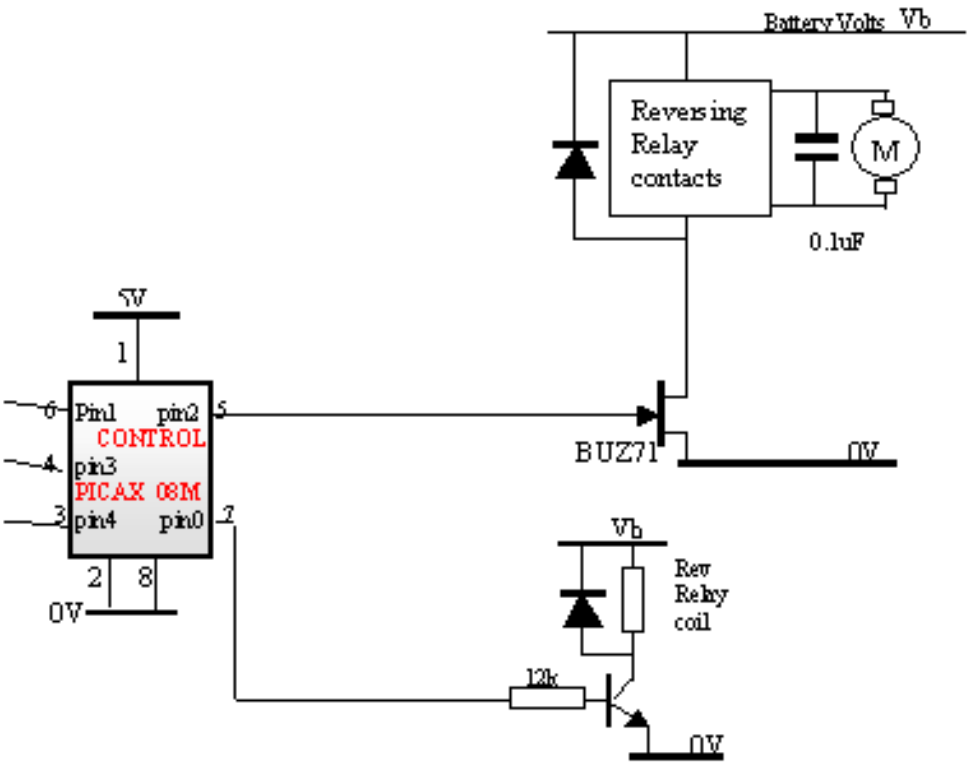
The SANDSTONE & TERMITE RAILWAY
"Linking the Outback to the World"

PICAXE TUTORIALS

5. Picaxe Motor speed & direction control

NOTE: All these programs are for battery operated locos.

The Picaxe is ideally suited to drive a FET (or MOSFET) to control motor speed using Pulse Width Modulation (PWM). The PWM from the Picaxe is just fed to the Gate of the FET.
To control the direction I use a relay driven from the Picaxe. Here's the basic circuit:-



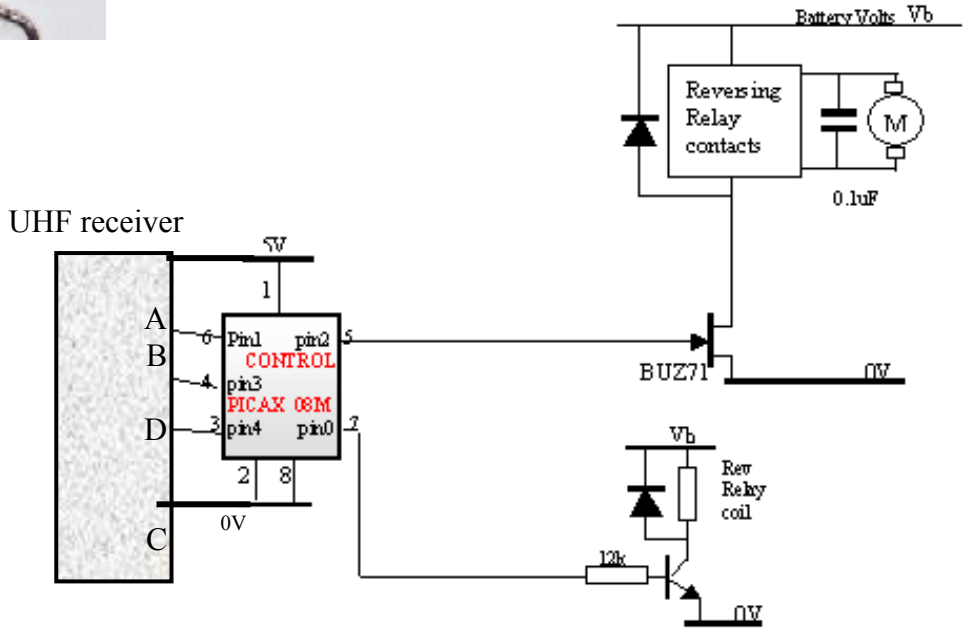
The Picaxe programming will depend on what kind of radio control you have – the circuit above just shows there are 3 available inputs to get the info into the Picaxe. eg accelerate, brake and toggle direction. all you need to do is decide in the program what to do and use the PWMOUT command. It runs in the background, so the program can go and check the state of the input signals while the motor speed remain constant.
NOTE: the PWMOUT command only works on pin2 (leg5) in the 8pin Picaxes.
The PWM frequency can be varied – see the Manual. I use 10kHz these days so I can't hear the motor whistle. There's a 'wizard' in the Picaxe editor to allow you to figure out the parameters you need, - this example is for a 4MHz picaxe:

PWMOUT 2, 99, pwmspeed
And pwmspeed will need to have a value of 0 (stopped) to 400 (full speed)

The direction relay is just pickedup or dropped out using the HIGH 0 or LOW 0 command, or you can use the **TOGGLE** command

The simplest speed control program

As a first example of a program, I use simple 'keyfob' type radio controllers. These have 4 outputs of which I use 3 – A for accelerate, B for Brake and D for toggle direction. The outputs are normally low and go to +5V when the transmitter button is pressed. So we just need a program to check the button state and adjust the motor speed and direction accordingly.
So I use pin1 for 'A', pin3 for 'B' and pin4 for 'D" in the circuit above. Pin2 is the speed output to the FET (it MUST be pin2 for the PWMOUT command !) and pin0 drives the reversing relay.



The basic program flow is like this:

SYMBOL PWMspeed=w1 this will be 0 to 400 using 10kHz PWM as above

Start: **PAUSE** 300 this value will determine how fast the train accelerates.
IF pin4=1 **THEN GOTO** direction: check direction first as it is most important.
IF pin1 =1 **THEN GOTO** Accel:
IF pin3=1 **THEN GOTO** brake:
GOTO start
Accel: **IF** PWMspeed>360 **THEN GOTO** topspeed: don't want to exceed 400
PWMspeed=PWMspeed+20 increase the speed
GOTO control:

```
Brake: IF pwmspeed<20 THEN GOTO stopped:          don't' want to have a negative speed!
       PWMspeed=PWMspeed-20          decrease the speed
       GOTO control:

Direction:          first we'll stop the motor, so this functions as an 'emergency stop' too.
       PWMOUT 2, OFF          stop the motor
       LOW 2                  be doubly sure!
       PWMspeed=0
       PAUSE 200              give motor a chance to stop
       TOGGLE 0               reverse the direction
D10:   IF pin4 =1 THEN GOTO D10          wait till release D button
       GOTO start


Topspeed:
       PWMOUT 2,OFF           don't bother with PWM at max speed, just turn hard on.
       HIGH 2                 full speed
       PWMspeed=400
       GOTO start

Stopped:
       PWMOUT 2,OFF
       LOW 2
       PWMspeed=0
       GOTO start

Control:            this just sends out the new speed to the motor.
       PWMOUT 2, 99,PWMspeed
       GOTO start
```

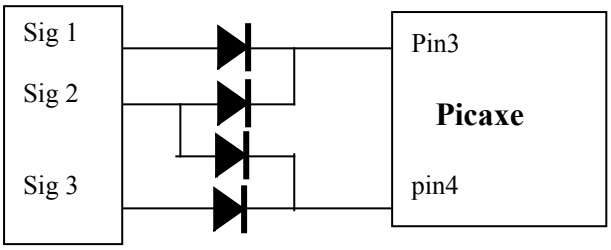
The value of the pause in line start: and the +/- 20 values in Accel: and Brake: will determine how fast the speed changes if you hold down the A or B buttons – ie like inertia. With the values shown, it will take 400/20 x .3 sec to reach max speed = 6 seconds.

Complete listing of:-

```
  simplest PWM 3button.bas
'example of the simplest motor speed control using 3 inputs from UHF Rx
'Greg Hunter    12/4/14                    84 bytes
'suitable for 08M  and 08M2   at 4 MHz

'pin0= reversing relay
'pin1= accelerate button input ('A' button)
'pin2= motor PWM output to FET gate
'pin3= brake button input
'pin4= toggle direction button input

'define variables:
SYMBOL PWMspeed=w1          'this will be 0 to 400 using 10kHz PWM
                            '(PWMOUT 2, 99,400max)

'The value of the pause in line start:  and the +/- 20 values in Accel: and Brake:
'will determine how fast the speed changes when you hold down the A or B buttons 'ie like inertia.
```

```
'With the values shown, it will take 400/20 x .3 sec to reach max speed = 6 seconds.
'there will be 400/20 'steps' of speed.

SYMBOL maxPWMsteps=400
'select the following values for suitable inertia using equation:
                        'time to reach max speed=maxPWMsteps/accBRsteps x pausetime
SYMBOL accBRsteps=20    'the number of different speed values outputted, from 0 to max
SYMBOL pausetime=300    'ms
'-----------------------------------------------------------------------
Start:
        Pause pausetime             'this value will determine how fast the train accelerates.
        If pin4=1 then direction:   'check direction first as it is most important.
        If pin1 =1 then Accel:
        If pin3=1 then brake:
        Goto start

Accel:
        if PWMspeed>360 then topspeed:          'don't want to exceed 400
        PWMspeed=PWMspeed+accBRsteps
        Goto control:

Brake:
        if pwmspeed<20 then stopped:            'don't' want to have a negative speed!
        PWMspeed=PWMspeed-accBRsteps
        Goto control:

Direction:          'first we'll stop the motor, so this functions as an 'emergency stop' too.
        PWMOUT 2, OFF           'stop the motor
        LOW 2                   'be doubly sure!
        PWMspeed=0
        PAUSE 200               'give motor a chance to stop
        Toggle 0                'reverse the direction
D10:    If pin4=1 then D10      'wait till release D button
        Goto start

Topspeed:
        PWMOUT 2,OFF            'don't bother with PWM at max speed, just turn hard on.
        HIGH 2                  'full speed
        PWMspeed=maxPWMsteps
        Goto start

Stopped:
        PWMOUT 2,OFF
        LOW 2
        PWMspeed=0
        Goto start

Control:            'this just sends out the new speed to the motor.
        PWMOUT 2, 99,PWMspeed
        Goto start
'++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

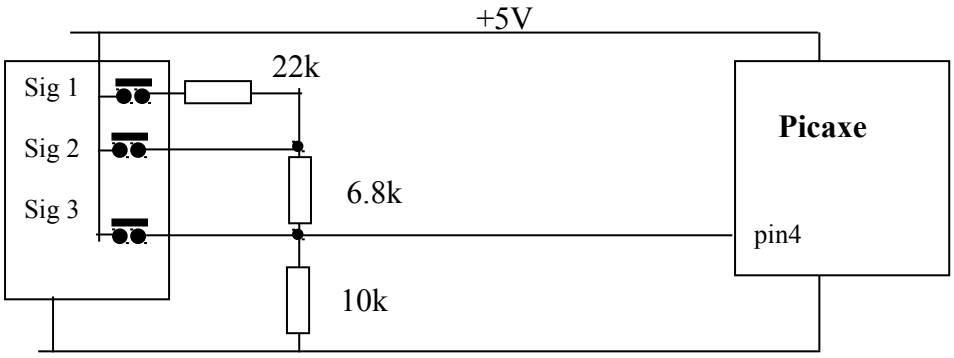
Multiplexing inputs using Diodes
So what do you do if you need more inputs that are available with the 8 pin version?
Well you can go to the 14 pin version. But it is possible to 'multiplex' inputs to get an extra 1 or 2 inputs.
Here's one way, using diodes. Say you have 3 input signals but only 2 input pins available and assuming only one input at a time is high.



And the program flow will look like this:-

IF pin3=1 and pin4=1 **THEN GOTO** process signal 2
IF pin3=1 **THEN GOTO** process signal 1
IF pin4=1 **THEN GOTO** process signal 3

Multiplexing inputs using analogues
Another way to multiplex is to use different analogue values to represent inputs. Again assuming only one signal is high at any one time.



When signal 1 is high the voltage on pin4 is 0.25 * 5V= 1.25V
When signal 2 is high. The voltage on pin4 is 0.6 *5V=3.0V
When signal 3 is high, pin4 is 5V

Now we choose voltage halfway between these as our 'discrimination' points =
0.6V (120 counts), 2.1V (430 counts) and 4.0V (820 counts)

So the program flow looks like:
READADC10 4, word
IF word<120 **THEN GOTO** process no inputs
IF word<430 **THEN GOTO** process input signal 1
IF word<820 **THEN GOTO** process input signal 2
process signal input 3

see tutorial 6. Advanced Picaxe motor control for continuation

Appendix: 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			