```
'*****************************************************************
'**                                                             **
'**            MD25 Motor Control Board with RD02 Drive System  **
'**       PICAXE 28X2 5V Module @8 MHz (AXE200) using i2c mode  **
'**                                                             **
'**            RD02 - 100 mm dia wheels; max speed 3.3 RPS      **
'**                                                             **
'**              Copyright 2011 - Middlesex University         **
'**            Commercial use of this software is prohibited    **
'**            Private and educational use only is permitted    **
'**                                                             **
'**             Written by Dr Stephen Prior - March 2011        **
'**     http://www.robot-electronics.co.uk/htm/picaxe_examples.htm **
'**                                                             **
'*****************************************************************
```

```
hi2csetup i2cmaster, $b0, i2cslow_8, i2cbyte
hi2cout 16, (32)
hi2cout 16, (50)
hi2cout 0, (128, 128)
pause 1000
let b18 = 255
```

```
'Communicate with MD25 using i2c, default address b0.
'Resets the encoder registers to zero.
'Disables 2 sec timeout of motors (V. 2 onwards); by default this is on.
'Stop both motors.
'Pause for 1 sec to ensure both wheels are fully stopped.
'Set the initial wheel speed.
'(max = 255; min = 129 for forward motion; max = 0; min = 127 for backward motion).
```

```
main:
hi2cout 15, (0)                                  'Set mode of operation, default is 0 (options are 0, 1, 2 or 3).
hi2cout 14, (10)                                 'Set the Acceleration register (1-10) (10 = 0.65 sec full speed to full stop).
hi2cout 0, (b18,b18)                             'Set Motor1 speed (mode 0,1) or speed (mode 2,3).
                                                 'Speed range of 0 (full reverse), 128 (stop), 255 (full forward).
gosub EncoderRead                                'Jump to the encoder read subroutine.
let W4 = W1-W3                                   'Capture and compare encoder count error (W3=b7:b6) & (W1=b3:b2) (Assuming less
than 65535).


'sertxd ("W1 = ",#W1, 1,"W3 = ",#W3, 1, "Diff =",#W4, CR, LF)   'Serial transmit the range data (W1, W3 and Difference W4) to the monitor.
W7 = 1000                                        'Target encoder count (360 counts = 0.314 m for a 100 mm diameter wheel).
let W8 = W7-W1                                   'Calculate the error (W8) between the target (W7) and encoder 1 (W1).
if W8 <= b18 then Slowdown                       'If the error is less than or equal to the initial set speed, then invoke the slowdown
subroutine.
sertxd ("W1 = ",#W1, 1,"W3 = ",#W3, 1, "W8 =",#W8, CR, LF)      'Serial transmit the range data (W1, W3 and Difference W8) to the monitor.
if W1 >= W7 or W3 >= W7 then Endex               'If either of the encoder counts is more than the set point then stop the wheels and
end.
'hi2cin 10, (b10)                                'Read the Supply Battery Voltage (121 = 12.1 Vdc).
'hi2cin 11, (b11)                                'Read Motor 1 Current (25 = 2.5 A).
```

```
'hi2cin 12, (b12)                                          'Read Motor 2 Current (25 = 2.5 A).
'sertxd ("Supply Voltage = ",#b10, 1, CR, LF)              'Serial transmit the Supply Battery Voltage (121 = 12.1 Vdc) to the monitor.
'sertxd ("M1Current = ",#b11, 1,"M2Current = ",#b12, 1, CR, LF)   'Serial transmit Motor Currents (25 = 2.5 A) to the monitor.
'pause 1000                                                'Pause 1 sec.
goto main                                                  'Go to main.
Slowdown:                                                  'Slowdown subroutine (effectively slows the motors as they approach the target).
let b18 = W8 MIN 140                                       'Let the speed variable (b18) equal the error variable (W8). Set a minimum speed value
of 140.
goto main                                                  'Go to main.
EncoderRead:                                               'Read each encoder into a 4byte variable (highest byte first).
hi2cin 6,(b5)                                              'Enc 2a - Encoder 2 Position, 1st byte (highest).
hi2cin 7,(b4)                                              'Enc 2b - Encoder 2 Position, 2nd byte.
hi2cin 8,(b7)                                              'Enc 2c - Encoder 2 Position, 3rd byte.
hi2cin 9,(b6)                                              'Enc 2d - Encoder 2 Position, 4th byte (lowest).
                                                                'Look to W3 for encoder position (up to 65535 counts).
hi2cin 2,(b1)                                              'Enc 1a - Encoder 1 Position, 1st byte (highest).
hi2cin 3,(b0)                                              'Enc 1b - Encoder 1 Position, 2nd byte.
hi2cin 4,(b3)                                              'Enc 1c - Encoder 1 Position, 3rd byte.
```

```
hi2cin 5,(b2)                                   'Enc 1d - Encoder 1 Position, 4th byte (lowest).
                                                        'Look to W1 for encoder position (up to 65535 counts).

return                                          'Return from subroutine.

Endex:                                          'Endex subroutine (stops the motors once the target is reached).
hi2cout 0, (128, 128)                           'Stop both motors.
gosub EncoderRead                               'Jump to the encoder read subroutine.
sertxd ("W1 = ", #W1, 1, "W3 = ",#W3, 1, CR, LF)  'Serial transmit the final encoder count data (W1 & W3) to the monitor.
pause 500                                       'Pause for 0.5 sec.
gosub EncoderRead                               'Jump to the encoder read subroutine to double check encoder counts.
sertxd ("W1 = ", #W1, 1, "W3 = ",#W3, 1, CR, LF)  'Serial transmit the final encoder count data (W1 & W3) to the monitor.
hi2cout 16, (32)                                'Resets the encoder registers to zero.
end
```