

Filename: c:\documents and settings\bruce\my documents\documents\projects\picaxe\listerpica
Date: 11/3/2009

```
#picaxe 40x1
' Written Sept 2007 RB McCreary
' Revision Log:
' 11/15/07 Servo reversing via input 7 added
' 11/23/07 Added tachometer calibration test via input 6
' 11/29/07 Fix tach calibration mode to set comm status bit OK, pause longer for
'          gen on/off, add delay for servo power on
' 12/01/07 modify initialization routine to turn generator off a 2nd time, revise servo
'          limits in compressor_on and compressor_off, change tach calibration test
'          code in get_sw_pkt to operate on idle timeout switch.

'Functional Description of the Program:
'
'A 40x1 Picaxe microcontroller controls the starting and stopping of the Lister
'(oid) 6/1 'diesel engine, a 1929 design with external flywheels. Starting is via
'Gast 4AM air motor with a rubber drive wheel, held against the flywheel by an
' air cylinder. Fuel injection rack closer and exhaust valve lifter (for compression
'release) are also air cylinders. A glow plug from Utterpower is activated for starting
' when temperature is low.

'The Lister (oid) is also instrumented with a gear tooth counter (hall effect) on one
'cast iron flywheel, an over temperature switch on the cylinder head, a Murphey
'vibration switch bolted to the cylinder, and magnetic reed relays to sense a magnet
' floating in an oil level sight tube. Idle will be detected by non-contact magnetic
' throttle sensor to allow auto shut down when extended idle (10 minutes) occurs,
' unless the idle timeout over-ride switch is on.

'Engine room sensors include a pressure switch for low cranking air pressure, and
' an analog temperature sensor. Futures will include another hall effect gear tooth
'sensor for detecting the air compressor being belted (spinning).

' The Listeroid drives a ST3 3KW generator head and a 5 Hp, 3 cylinder, 2 stage
' air compressor from Eaton Compressors. The "bad voltage" signal is a status line
' to indicate that an extended over/under-voltage situation has been detected by the
' external generator regulator, and that the load has been automatically disconnected
' to avoid damage to equipment. The generator can be connected or disconnected
' from the AC power line from the picaxe via a pulse to the latching relay set or reset
' signals.

' The air compressor is controlled through an RC servo controlled air switch which
' allows the air compressor unloader valves (three) to be pressurized in order to make
' the compressor "free wheel". This allows the unloaded air compressor to be belted
' while larger AC loads are being used. With the switch in the "on" position, a standard
' compressor pilot valve controls the unloader valve, turning off the compressor when
' 150 psi is achieved. When the air compressor is not belted, a ball valve is manually
'closed to close off the unloader/pilot valves.

' For starting and control, a local (HOL) 30 gallon air tank is fed via a check valve. If
' the shop 500 gallon tank is low, this tank will still provide more than 10 starts plus
' control. The controlling program will not allow the engine to remote start if this local
' tank is also getting low. (Pressure switch sensor.) Full manual (hand crank) start and
' operation is allowed, for both compressor and generator with not even a battery
' required.

' The Picaxe 40x1 single chip microcontroller resides in the "House of Lister"
'(aka engine room), and receives the control switch settings from the
' remote shop control/status panel via an asynchronous serial data link (opto-isolated
' current loop physically). The Lister Status data sent is displayed via
' LEDs at the remote shop control/status display.

'House of Lister
'Picaxe 40X1 Pin Out Assignments and symbol names

'Outputs:
'Pin#    Port    Pin#    Function    Symbol    Text Replacement
'-----
'33      default 0    vents open    vent_open    0
'34      default 1    vents close    vent_close    1
```

(continued...)

```
'35      default 2      rack close      rack_close      2
'36      default 3      starter engage      start_engage      3
'37      default 4      starter motor      start_motor      4
'38      default 5      air valve open      air_valve_open      5
'39      default 6      air valve pwr on      air_valve_pwr      6
'40      default 7      compressor valve servo      compv_servo      7

'
'15      port c      0      generator relay off      (no symbol)
'16      port c      1      generator relay reset(on)      "      portc 1
'17      port c      2      comp servo pwr      "      portc 2
'18      port c      3      glow plug pwr      "      portc 3
```

'Symbol table for Output Pins

```
'-----
symbol vent_open =      0
symbol vent_close =      1
symbol rack_close =      2
symbol start_engage =      3
symbol start_motor =      4
symbol air_valve_open =      5
symbol air_valve_pwr =      6
symbol compv_servo =      7
'
```

'Inputs:

Pin#	Port-	Pin#	Function	Symbol	Text Replacement
'19	default 0	oil emergency	oil_em	pin0	
'20	default 1	temperature or vib-			
		ration emergency		temp_vib_em	pin1
'21	default	2 volts_bad		volts_bad	
'22	default	3 low starting air			
		pressure		low_start_air	pin3
'27	default	4 tachometer		tach	
'28	default	5 idle timeout		idle_timeout	
'29	default 6	spare			
'30	default	7 spare			
'8	ADC 5	temperature			
'9	ADC 6	battery voltage			
'10	ADC 7	spare			
'15-17	port c	input or output spares			
'23-24	port c	input or output spares (23 is 12c sda)			

'Symbol table for Input Pins

```
'-----
symbol oil_em =      pin0
symbol temp_vib_em =      pin1
symbol volts_bad =      pin2
symbol low_start_air =      pin3
symbol tach =      4
symbol idle_timeout =      pin5
```

' Other pin definitions:

```
'1      reset
'2-5      port A inputs or ADC (spares)
'6-7      serial download pins (reserved)
'12,31      0 volts(gnd)
'11,32      + volts (5 volts)
'25      hardware serial TX to communicate with remote 40x1
'26      hardware serial RX to communicate with remote 40x1
```

'see page 24 of the datasheet ("Getting Started") for complete pinout of the 40x'

```
'Lister Status Bytes 1-3. These bytes sent to the remote shop 40x1
'processor which displays status information and sends a switch packet every second. It
' is also used for internal status keeping.
```

```
symbol lstat1 = b8      'lstat1-3 will be used here after to refer to these
symbol lstat2 = b9      'three status bytes
symbol lstat3 = b10
```

' bit 7	0 valid status byte tag
' bit 6	engine starting
' bit 5	engine running
' bit 4	engine shutting down
' bit 3	compressor on
' bit 2	generator on
' bit 1	vents open
' bit 0	idle timeout override on

```
' bit 7      0 valid byte tag
' bit 6      compressed air on (motorized ball valve open)
' bit 5      spare
' bit 4      spare
' bit 3      spare
' bit 2      spare
' bit 1      spare
' bit 0      spare
```

```
'bit 7      0 valid byte tag bit
'bit 6      last switch byte packet received OK
'bit 5      spare
'bit 4      spare
'bit 3      status code bit 3 (msb)
'bit 2      status code bit 2
'bit 1      status code bit 1
'bit 0      status code bit 0 (lsb)
```

```
'0000 all is well
'0001 idle timeout
'0010 vibration or over-temperature emergency
'0011 oil level emergency
'0100 generator voltage emergency
'0101 generator frequency emergency
'0110 low air emergency
'0111 overspeed emergency
'1000 underspeed emergency
'1001 failed start- too slow cranking speed
'1010 failed start- no ignition
'1011 failed start- didn't come up to speed
'1100-1111 spare
```

```
symbol start_speed= 11
symbol ign_speed= 24
symbol norm_min_speed= 60
symbol abs_min_speed= 50
symbol norm_max_speed= 80
symbol abs_max_speed= 90
```

(continued...)

```
symbol underspeed_count = b17
symbol overspeed_count= b18
```

'Operating State Flag Bytes:

'-----

```
symbol comp_on=b12          'compressor on=1 else 0
symbol idle_to_flg=b13      'idle timeout on=1else 0
symbol em_shut_down_flg=b20 'engine has had em shut down
```

'sw_pkt is the current control switch position packet byte sent by asynchronous
'serial data from the remote control panel. This occurs in the background, via hw
'serial port and an interrupt routine (picaxe supplied polling interrupt).

```
symbol sw_pkt=b16          'bit 0 engine on/off
'                          bit 1 generator on/off
'                          bit 2 compressor on/off
'                          bit 3 idle timeout on/off
'                          bit 4 vent open/auto
'                          bit 5 spare
'                          bit 6 spare
'                          bit 7 - 0- valid data tag (can't ever be $FF)
```

'-----
'Initializations

```
'      enter on power on or processor reset only
'      start with everything in a known position
'      main air ball valve, compressor servo/valve, vents
```

'-----

main:

```
' set all outputs low
'   pins =0
'   pinsc=0
```

```
' bug in 40x1 rev A.0 firmware glitches portC pins on during power on initialization
' so the latching relay to the generator gets turned on. Now make sure it is turned off.
```

```
      gosub generator_off
```

```
' set up serial data link
'   hsersetup b600_4, %10      '600 baud at 4MHz, inv output, no background inp
```

```
' initialize data
'   lstat1=0
'   lstat2=0
'   lstat3=0
'   comp_on=0
'   idle_to_flg=0
'   em_shut_down_flg=0
```

```
' get engine rpm
'   gosub lister_tach          'returns tach count in b0
'   if b0 = 0 then 'engine off
'   gosub air_off
'   goto off_state
'   endif

'   if b0>= norm_min_speed then      'running speed
'   gosub air_on
'   sw_pkt=%00000001      'set run bit of sw_pkt in case remote is off
'   lstat1=%00100000      'set run bit of lstat 1
'   goto run_state
'   endif
```

(continued...)

```
' not stopped but not fully running- shut down
  gosub air_on
  gosub shut_down
  gosub air_off
' continue to off state
```

```
'-----
'Off State
```

```
'entered- via successful shut down or if processor switched on and engine is off
'exits- a successful start will result in exit to the Run State
```

```
'
'tasks- Open or close vents according to vent open override switch. Normally
'       vents are closed in the off state.
'       Start engine via the start_eng subroutine if start switch is set.
```

```
'note- to programmer; don't clear emergency code in lstat3 until start is selected
'or vents are switched (some valid switch action) so that an emergency shut down
' code can be seen.
```

```
' The shut_down_flag indicates that an emergency shut down has happened. In this case
' the run switch is ignored until it is cycled off, then back on.
```

```
'-----
off_state:
```

```
    gosub get_sw_pkt          'send listerstatus, get sw_pkt via asych serial link
    b0=sw_pkt
'    note- sw_pkt is not updated unless remote control unit is powered on.

'    set the idle timeout flag
    idle_to_flg=bit3

'    check for start switch off (bit 0 of sw_pkt)
    if bit0=0 then
        em_shut_down_flg=0      'reset em shut down flag when run/off switch set to off
        goto vent_ck           'no start request
    endif

'    start requested
    if em_shut_down_flg=1 then vent_ck      'no start until switch set to off then back on
    gosub check_em 'check for emergencies, return code in b0
    if b0=0 then start_engine

'    no start allowed due to low air or emergency condition
    em_shut_down_flg=1
    lstat3=lstat3 and %11110000      'clear code bits in lstat3
    lstat3=lstat3 or b0              'put error code number in lstat3
    goto off_state
```

```
vent_ck:
```

```
    b0=sw_pkt
    b1=lstat1

    if bit4=1 and bit9=0 then      'vents open switch on and vents are closed, so open ven
        gosub air_on
        gosub vents_open
        gosub air_off      'debateable - also perhaps add an input to sence the valve pos???
    endif

    if bit4=0 and bit9=1 then      'vents open switch off and vents are open, so close ven
        gosub air_on
        gosub vents_close
```

(continued...)

```
        gosub air_off
    endif

'confirm engine off here???

    goto off_state

start_engine:
    lstat3=lstat3 and %11110000      'clear error codes in last 4 bits
    gosub start_eng 'returns b=0 if good start
    lstat3=lstat3 or b0      'put returned start error code (0 if good) in lstat3
    if b0=0 then run_state

'    bad start here

    gosub shut_down
    gosub air_off
    em_shut_down_flg=1      'no restart until start switch is flipped off then back on
    goto off_state

'-----
'Run State

' entered- via successful start or if processor switched on with engine running
' exits - emergency off or switched off

'tasks-
' monitor engine and system for emergencies
' process switch commands- off, compressor, generator, idle override
' check for idle timeout unless override is on
'-----

run_state:
    underspeed_count=0
    overspeed_count=0

run_loop:
'    first check for emergencies
'-----
        gosub check_em      'returns 0 if no emergencies
        if b0!=0 then disaster      'disaster will put returned error code (b0) in lstat3

        gosub lister_tach      'returns count of spoke sensor in b0
        if b0 < norm_max_speed then ck_low_speed

'    max speed exceded
        if b0 > abs_max_speed then
            b0=7 'code 7 is overspeed
            goto disaster
        endif

'    minor overspeed detected- let's see if it's temporary
        inc overspeed_count
        if overspeed_count = 8 then      '??? how many loops before overspeed timeout???
            b0=7 'code 7 is overspeed
            goto disaster
        endif

ck_low_speed:
    if b0 > norm_min_speed then

'    normal speed detected- clear under/overspeed counters, continue with run_state work

        underspeed_count=0
        overspeed_count=0
        goto process_switches
```

(continued...)

```

endif

' underspeed detected-
  if b0 < abs_min_speed then
    b0=8 'code 8 is underspeed
    goto disaster
  endif

' minor underspeed- see if it continues
  inc underspeed_count
  if underspeed_count=5 then
    b0=8 'code 8 is underspeed
    goto disaster
  endif

process_switches:
  gosub get_sw_pkt 'send listerstatus, get sw_pkt via asych serial link
  -----
  b0=sw_pkt 'b0 gets the switch packet so we can play with it's bit
            variables bit0 thru bit7

' first check for switch command to off state
  if bit0=0 then
    em_shut_down_flg=0 'normal shut down, this is not set
    gosub shut_down 'shut down and got to off_state if off commanded
    goto off_state
  endif

' if switch says on and generator is presently off- turn on gen output power relay
  bl=lstat1 'bit 2 of lstat 1 is now bit 10
  if bit1=1 and bit10=0 then gosub generator_on

' if switch says off and generator is presently on-
  if bit1=0 and bit10=1 then gosub generator_off

' now for the compressor switch
' if compressor switch says on but compressor state is off, turn it on via servo
' controlled air switch to the pilot/unloader valves

  if bit2=1 and comp_on=0 then gosub compressor_on

' turn off compressor if switch says off and compressor is on

  if bit2=0 and comp_on=1 then gosub compressor_off

' next is idle time out - but since I don't have the throttle sensor
' yet I'm going to skip it

  goto run_loop 'do it all again - stay in this run_state loop
                'until exit conditions are met

disaster:
  lstat3=lstat3 and %11110000 'strip off low bits
  lstat3=lstat3 or b0 'over/underspeed code in b0
  em_shut_down_flg=1
  gosub shut_down
  gosub air_off
  goto off_state

'-----
'Subroutine start_eng Starts the Lister!

' called by off state only if all systems precheck OK
' returns b0 = 0 success
'             %1001 too slow cranking speed
'             %1010 no ignition
'             %1011 didn't come up to speed

```

(continued...)

```
' tasks-
'      operate all the starting valves, servo and hardware with finesse and care
'-----
start_eng:
'      open motorized ball valve for air supply
'          gosub air_on      '4 seconds

'tell remote air is on
'          gosub get_sw_pkt
'
'      future temperature check here for auto glow
'      turn on glow plug
'          high portc 3

'      open vents, close rack, engage starter
'          high rack_close
'          high start_engage
'          gosub vents_open      '4 seconds
'          lstat1=%01000010      'everything in a known state- all off but vents and starting bi

'      tell remote that vents are open, starting
'          gosub get_sw_pkt

'      starting the Lister now!
'          high start_motor

'      now wait for speed to come up
'          for b4=1 to 8
'              gosub lister_tach
'              if b0 > start_speed then start_spd
'          next b4

'      failed start speed too slow
'          gosub stop_crank      'turns off glow plug too
'          gosub shut_down
'          b0=%00001001
'          return

start_spd:
'      low rack_close      ' release decompression and inject fuel

'      for b4 = 1 to 7      '7 seconds to ignite and run
'          gosub lister_tach
'          if b0 > ign_speed then run_spd
'      next b4

'      failed start
'          gosub stop_crank
'          gosub shut_down
'          b0=%00001010
'          return

run_spd:
'      low start_motor      'stop air motor stater
'      low start_engage

'      for b4=1 to 10      '10 seconds to get up to minimum running speed
'          gosub lister_tach

'          if b0> norm_min_speed then
'              low portc 3      'glow plug off
'              b0=0
'              lstat1=%00100010      'running and vents open!
'              return
```


(continued...)

```
endif

next b4

' engine died after ignition- turn off glow plug and shut down
  low portc 3
  gosub shut_down
  b0=3
  return

'-----
'Subroutine shut_down
' Indicate shutting down, turn off generator, compressor, close rack/decompression,
' wait for engine to spin down. Close vents unless vent override switch is on.
' turns off main air valve on completion.
'-----
shut_down:
  high rack_close ;close the fuel rack and decompress via air cylinders
' clear all lstat but vents (1) and shut down bit(4)
  lstat1=%00010010

'turn off generator
' always do it since it's fast
  gosub generator_off

' tell remote we're shutting down
  gosub get_sw_pkt

' then turn off compressor- just in case it's on
  gosub compressor_off

' now wait for engine to coast to a stop
coast1:
  pause 2000
  gosub lister_tach
  if b0 >1 then coast1 'note will hang here forever if shut down fails???

' normal stop here
  low rack_close
  gosub vents_close
  gosub air_off 'note this routine clears the air on bit in lstat2
  lstat1=lstat1 and %00000010 'clear all but vents open
  return

'-----
'Subroutine check_em

'Engine and system emergency check subroutine, sets lstat3 code,
' returns bo=error code if emergency condition detected
'-----

'Emergency codes-
'-----
'1 low/high oil
'2 over temperature/vibration
'3 engine speed too high/low
'4 low air pressure (impending loss of engine control)
```

check_em:

(continued...)

```
'  check oil first
if oil_em = 1 then
    b0=3
    return
endif

'  now check for excess vibration or head temperature too high
if temp_vib_em = 1 then
    b0=2
    return
endif

'  now low air check
if low_start_air=1 then
    b0=6
    return
endif

'  all is OK-
    b0=0
    return
```

```
'-----
'Stop Cranking Engine subroutine-          stop_crank
'      disengages and turns off starter air motor, turns off glow plug
'-----
```

```
stop_crank:
    low portc 3      'turn off glow plug
    low start_motor
    low start_engage
    return
```

```
'-----
'Lister_tach subroutine-          lister_tach
'      returns 1 second count of gear tooth sensor in b0
'      this is a subroutine to facilitate modification to ADC input for bench testing
'-----
```

```
lister_tach:

    count tach, 1000, w0      'since count is less than 255, this should do it
    b19=w0      'test value to be displayed on remote
    return
```

```
'-----
'Subroutine air_on
'      'turns on main air (motorized ball) valve
'-----
```

```
air_on:
    high air_valve_open      'direction signal to motorized ball valve relay
    high air_valve_pwr      'power on the motorized ball valve
    pause 2000
    lstat2=lstat2 or %01000000      'bit 6 is air on
    gosub get_sw_pkt
    pause 4000
    low air_valve_pwr
    low air_valve_open      'no need to burn current holding a relay for nothing

    return
```

(continued...)

```
'-----
'Subroutine air_off
'    turns off main air (motorized ball) valve
'-----
air_off:
    low air_valve_open
    high air_valve_pwr
    pause 2000
    lstat2=lstat2 and %00111111
    gosub get_sw_pkt
    pause 4000
    low air_valve_pwr
    return

'-----
'Subroutine generator_off
'    turns off generator output relay via pulse to a latching relay
'-----
generator_off:
    high portc 0
    pause 300
    low portc 0
    lstat1=lstat1 and %11111011          'clear bit 2 of lstat 1 for generator off flag
    return

'-----
'Subroutine generator_on
'    turns on generator by pulsing  reset pin on latch relay
'-----
generator_on:
    high portc 1
    pause 300
    low portc 1
    lstat1=lstat1 or %00000100          'set bit 2 of lstat1 for generator on flag
    return

'-----
'Subroutine compressor_on
'    fiddles with the RC servo to turn the air switch to the "load compressor"
'    position.  Silly if the compressor isn't belted up, of course.
'-----
compressor_on:
    high portc 2                      'turn on power to the RC servo
    pause 100
    servo compv_servo, 190          'first test with 220 caused overdrive/stalling
    pause 1500
    low compv_servo                  'turn off servo pulses
    low portc 2                      'turn off servo power
    comp_on=1
    lstat1=lstat1 or %00001000        'set compressor on bit in lstat1
    return

'-----
'Subroutine compressor_off
'    fiddles with the RC servo to turn the air switch to the "unload compressor"
'    position.  Silly if the compressor isn't belted up, of course.
'-----
compressor_off:
    high portc 2                      'turn on power to the RC servo
```

(continued...)

```

    pause 100
    servo compv_servo, 60    'first test with 80 was underdriven.
    pause 1500
    low compv_servo          'turn off servo pulses
    low portc 2              'turn off servo power
    comp_on=0
    lstat1=lstat1 and %11110111    'compressor bit in lstat1=1 for off
    return

```

```

'-----
'Subroutine vents_open
'  opens the vent doors in the House of Lister via pneumatic solenoid and
'  cylinders. The motorized air valve for system air pressure must be already
'  open.
'-----

```

```

vents_open:
    high vent_open
    pause 4000
    low vent_open
    lstat1=lstat1 OR %00000010    'bit 1 is the vent state- 1 open
    return

```

```

'-----
'Subroutine vents_close
'  closes the vent doors in the House of Lister via pneumatic solenoid and
'  cylinders. The motorized air valve for system air pressure must be already
'  open.
'-----

```

```

vents_close:
    b0=sw_pkt
    if bit4=1 then
        return    'ignore close request if vent open switch is set
    endif

    high vent_close
    pause 4000
    low vent_close
    lstat1=lstat1 AND %11111101    'lstat bit 1 is the vent state- 0 close
    return

```

```

'-----
'Subroutine get_sw_pkt
'  sends lstatus bytes and recieves sw_pkt from remote processor
'  must have hsersetup in init routine
'-----

```

```

get_sw_pkt:
'  first send out lstatus package with $FF end of data marker (4 bytes total)
'
'  test code- use idle timeout switch to enable tach count to remote display LEDs
    b0=sw_pkt
    if bit4=0 then    'idle timeout is bit 4 of the last sw_pkt recieved from remote
        hserout 0, ($FF, lstat1, lstat2, lstat3)
    else
        ' special calibration/test routine- if input pin6=1 then
        ' put tach digital value (byte) in place of Lister status info such that the
        ' high order nibble appears in the gen, comp, idle, and vents LEDs and
        ' the low order nibble appears in the 4 bit status code LEDs

        b21=b19 AND %00001111    'b19 is last tach value
        b21=b21 OR %01000000    '2nd msb is last packet OK bit
        b1=b19

```

(continued...)

```

        b0=0
        bit1=bit12      'vents open LED gets bit 12 or lsb of higher nibble (4 bits)
        bit0=bit13      'idle shutdown LED gets bit 13
        bit3=bit14      'compressor on LED gets bit 14
        bit2=bit15      'generator on LED gets bit 15 (msb)
        hserout 0, ($FF, b0, lstat2, b21)      'default outputs will get high order bi
    endif

'    now get new sw_pkt plus verification data
'    no break, 200ms timeout, stack address 0, 2 bytes

        hserin [200, ser_timeout], 0,2
        ptr=0
        b0=@ptrinc
        b1= @ptr
        b1= inv b1
        if b0 = b1 then good_pkt      'second byte should be inverse of sw_pkt
ser_timeout:
        lstat3=lstat3 and %10111111      'bit 6 of lstat3=0 says bad data
        return

good_pkt:
        sw_pkt=b0
        lstat3=lstat3 or %01000000
        return
```