```
' {$STAMP BS2}
' {$PBASIC 2.5}


' -----[ EEPROM Data ]----------------------------------------------------
'area of ROM reserved to hold compass calibration data
'this area is held at the top of the ROM to mitigate conflicts
'with ROM needed for the programs.  Programs use ROM from the
'bottom up.
CompassOffsets DATA @ 0, (4)              ' Stores x and y axis offsets
CompassLowVal  DATA     (1)               ' Stores index of lowest angle
CompassCal     DATA     (16)              ' 16 reference compass angles

'-----------------------------------------------------------------
'constants and variables for "Leader" program
irleft  VAR    Bit
irright VAR    Bit
pulsecount VAR Byte

'-----------------------------------------------------------------
'constants and variables for "Follower" program
kpl CON -35
kpr CON 35
setpoint CON   2
centerpulse  CON        750
```

```
freqselect   VAR        Nib
irfreq    VAR          Word    'double use with axisoffset
distanceleft VAR       Nib
distanceright VAR      Nib
pulseleft     VAR      Word    'double use with span
pulseright    VAR      Word    'double use with angleoffset



'-------------------------------------------------------------
'Constants and variables for TV remote control of the BOE-BOT.
'-------------------------------------------------------------
'This program uses the Sony code "0000".  Press keys keys 1-9
'to control BOT movement, press the vol -/+ and ch + keys
'to control the ATV camera if installed.  These codes can be
'confirmed by running the "pulsetobyte.bs2" program with the
'TV remote to be used.  Once you have verified the specific
'codes for the keys you wish to use, modifiy the constant
'statements below to match the codes.



 irDet  PIN 0            'IR control
 spkr   PIN 4            'all subroutines
 sel    PIN 7
 cw_id  PIN 6
 'TV remote key assignments
```

```
menu          CON   96
guide         CON   14
info          CON   58
exit_key      CON   99
volplus       CON   18
volminus      CON   19
chplus        CON   16
chminus       CON   17
enter         CON   20
power         CON   21
'the following additional constants are required if
'the menu key is pressed immediately before the listed
'key is pressed because the menu key places the TV remote
'into a different control mode for a short period of time
menu_chplus    CON  116
menu_volminus  CON  52
menu_volplus   CON  51
menu_chminus   CON  117
'constants used for generating the Morse code ID for ATV operation

tone      CON 1800      'Morse tone frequency
dit_time  CON 50        'dit time unit
bit_space CON dit_time*2'bit spacing is dah unit, since on dit time is at
                        'end of each dit or dah, need only 2 more
dah_time  CON dit_time*3'dah time unit is 3 times dit time
```

```
'variables
irpulse       VAR   Word
remotecode    VAR   Byte
pulsecode     VAR   Word

'-------------------------------------------------------
'variables and constants for the temperature subroutines

'temperature test program

dq  CON 15
clk CON 5
rst CON 1
dsdata  VAR  Word           'double with light
sign  VAR dsdata.BIT8
t_sign  VAR Bit

rconfig CON $ac
wconfig CON $0c
cpu CON %10
nocpu CON %00
oneshot CON %01
cont  CON %00
startc  CON $ee
stopc CON $22
```

```
rtemp CON $aa
rhit  CON $a1
whit  CON $01
rlot  CON $a2
wlot  CON $02

' -----[ Pins/Constants/Variables ]-------------------------------------------
DinDout       PIN     15              ' P6 transceives to/from Din/Dout
En            PIN     1               ' P4 controls HM55B's /EN(ABLE)

Reset         CON     %0000           ' Reset command for HM55B
Measure       CON     %1000           ' Start measurement command
Report        CON     %1100           ' Get status/axis values command
Ready         CON     %1100           ' 11 -> Done, 00 -> no errors
NegMask       CON     %1111100000000000   ' For 11-bit negative to 16-bits

x             VAR     Word            ' x-axis data
y             VAR     Word            ' y-axis data
status        VAR     Nib             ' Status flags
angle         VAR     Word            ' Store angle measurement

'-----------------------------------------------------------------------------

current       CON     0               ' Table array index
previous      CON     1               ' Table array index
```

```
' -----[ Variables ]----------------------------------------------------------

axisOffset     VAR      irfreq              ' Axis offset


index          VAR      Status              ' EEPROM index

table          VAR      Byte(2)             ' Stores EEPROM table values

span           VAR      pulseleft           ' Span between table entries

angleOffset    VAR      pulseright          ' Offset btwn measured ANd table


spin       CON  16                     'data line out to computer

baud       CON  84                      '9600, 8-N-1

tpin      CON  11                      'data line to transmitter

baudmode  CON  813                        'to set 1200 baud according to formula

                                  'no add for polarity  see page 81 of bstamp

'-------------------------------------------------------------------------

'light measurement variableS


light  VAR pulsecode

'-------------------------------------------------------------------------

'variables used by mars subroutine

forwardflag     VAR Bit

distance_sign   VAR Nib

distance  VAR   pulseleft

'-----------------------------------------------

'initialized temperature
```

```
LOW rst
HIGH  clk
PAUSE 100
HIGH  rst
SHIFTOUT  dq,clk,LSBFIRST,[wconfig,cpu+cont]
LOW rst
PAUSE 50
HIGH  rst
SHIFTOUT  dq,clk,LSBFIRST,[startc]
LOW rst
'-------------------------------------------------

 main:

  HIGH sel
  PAUSE 10
  RCTIME sel,1,light
  IF light > 95 THEN
    GOSUB dit
    GOTO  mars

  ELSEIF light > 80 THEN
    GOSUB dit
    GOSUB dit
```

```
  GOSUB dit
  GOTO  follower

 ELSEIF light > 65 THEN
  GOSUB dit
  GOSUB dit
  GOTO  tv_remote_control

 ELSE

  GOSUB dit
  GOSUB dit
  GOSUB dit
  GOSUB dit
  GOTO leader
 ENDIF

END
'Suroutines----------------------------------------------------------
 send_start_tone:
   FREQOUT spkr,500,2000
 RETURN  'send_start_tone

 get_obstructions:
  FREQOUT 8,1,38500
```

```
  irleft=IN9
  FREQOUT 2,1,38500
  irright=IN0
 RETURN  'get_obstructions
'------------------------------------------------------------
mars:

 GOSUB get_compass

DO
 GOSUB get_obstructions
 IF (irleft=1) AND (irright=1) THEN
    DO WHILE (irleft=1) AND (irright=1)
    GOSUB get_obstructions
    distance=distance+1
    forwardflag=1
    PULSOUT 13, 800
    PULSOUT 12, 700
    PAUSE 20
    LOOP
 ELSEIF (irleft=0) AND (irright=1) THEN
    IF forwardflag THEN GOSUB send_data
    DO WHILE (irleft=0) AND (irright=1)
      GOSUB get_obstructions
      PULSOUT 13, 800
```

```
      PULSOUT 12,800
      PAUSE 20
    LOOP
    GOSUB get_compass
  ELSEIF (irleft=1) AND (irright=0) THEN
    IF forwardflag THEN GOSUB send_data
    DO WHILE (irleft=1) AND (irright=0)
      GOSUB get_obstructions
      PULSOUT 13, 700
      PULSOUT 12, 700
      PAUSE 20
    LOOP
    GOSUB get_compass
  ELSEIF (irleft=0) AND (irright=0) THEN
    IF forwardflag THEN GOSUB send_data
    GOSUB turn_right
    GOSUB turn_right
    GOSUB get_compass
  ENDIF
LOOP

RETURN  'mars
 '-------------------------------------------------------------
 leader:
```

```
DO
  GOSUB get_obstructions

  IF (irleft=0) AND (irright=0) THEN
    GOSUB back_up
    GOSUB turn_left
    GOSUB turn_left
  ELSEIF (irleft=0) THEN
    GOSUB back_up
    GOSUB turn_right
  ELSEIF (irright=0) THEN
    GOSUB back_up
    GOSUB turn_left
  ELSE
    GOSUB forward_pulse
  ENDIF
LOOP

'Leader subroutines-------------------------------------------
  forward_pulse:
    PULSOUT 13,800
    PULSOUT 12,700
    PAUSE 20
  RETURN                'forward_pulse
```

```
turn_left:
  FOR pulsecount=0 TO 20
    PULSOUT 13,700
    PULSOUT 12,700
    PAUSE 20
  NEXT
RETURN                'turn_left

 turn_right:
  FOR pulsecount=0 TO 20
    PULSOUT 13,800
    PULSOUT 12,800
    PAUSE 20
  NEXT
RETURN                'turn_right

back_up:
  PAUSE 300           'pause to prevent poping motion
  FOR pulsecount=0 TO 40
    PULSOUT 13,700
    PULSOUT 12,800
    PAUSE 20
  NEXT
RETURN                'back_up
'end leader program-----------------------------------------
```

```
'follower program-----------------------------------------
follower:

DO
 GOSUB get_ir_distances
 pulseleft=setpoint-distanceleft*kpl+centerpulse
 pulseright=setpoint-distanceright*kpr+centerpulse

 GOSUB send_pulse
LOOP

'follower subroutines--------------------------------------
get_ir_distances:
 distanceleft=0
 distanceright=0
 FOR freqselect = 0 TO 4
  LOOKUP freqselect, [37500,38250,39500,40500,41500],irfreq

  FREQOUT 8,1,irfreq
  irleft=IN9
  distanceleft=distanceleft+irleft
  FREQOUT 2,1,irfreq
  irright=IN0
  distanceright=distanceright+irright
```

```
  NEXT
 RETURN                  'get_ir_distances

 send_pulse:
  PULSOUT 13, pulseleft
  PULSOUT 12, pulseright
  PAUSE 5
 RETURN                  'send_pulse

'-------------------------------------------------------------
TV_remote_control:

pulsecode=750               'center AVT servo
DO
 GOSUB get_ir_remotecode

 SELECT remotecode
  CASE  1                'key 2 forward
    PULSOUT 13, 850
    PULSOUT 12, 650
  CASE  3                'key 4 left
    PULSOUT 13, 650
    PULSOUT 12, 650
  CASE  5                'key 6 right
    PULSOUT 13, 850
```

```
  PULSOUT 12, 850
CASE  7                   'key 8 backward
  PULSOUT 13, 650
  PULSOUT 12, 850
CASE  0, menu             'key 1 half left
  PULSOUT 13, 750
  PULSOUT 12, 650
CASE  2, guide            'key 3 half right
  PULSOUT 13, 850
  PULSOUT 12, 750
CASE  6, info             'key 7 half left rear
  PULSOUT 13, 750
  PULSOUT 12, 850
CASE  8, exit_key         'key 9 half right rear
  PULSOUT 13, 650
  PULSOUT 12, 750
CASE  4, enter            'key 5 halt
  PULSOUT 13, 750
  PULSOUT 12, 750
CASE volminus,menu_volplus    'camera left
  pulsecode=pulsecode+25
    IF pulsecode>1250 THEN pulsecode=1250
  PULSOUT 14, pulsecode
CASE chplus,menu_chplus       'center Camera servo
  FOR irpulse=0 TO 50
```

```
    PULSOUT 14, 750
    PAUSE 20
    NEXT
    pulsecode=750
  CASE volplus,menu_volminus     'camera right
    pulsecode=pulsecode-25
      IF pulsecode<250 THEN pulsecode=250
    PULSOUT 14, pulsecode
  ENDSELECT
LOOP

get_ir_remotecode:

 get_pulses:
  remotecode=0                'clear received code

 DO

  ' IF IN6 THEN GOSUB send_CW_ID   'check for ID_send signal from the ID timer IC
                        'if a high is present, it is time to send the
                        'ATV station ID

                         'waits until the long
  RCTIME 3,1, irpulse           'pause between repeating codes are sent
```

```
 LOOP UNTIL irpulse>1000

 RCTIME 3,0,irpulse              'trap for erronious start bit
  IF irpulse>1125 OR irpulse<675 THEN GOTO get_pulses

 RCTIME 3,0,irpulse              'getting data bits
  IF irpulse > 300 THEN remotecode.BIT0=1
 RCTIME 3,0,irpulse
  IF irpulse > 300 THEN remotecode.BIT1=1
 RCTIME 3,0,irpulse
  IF irpulse > 300 THEN remotecode.BIT2=1
 RCTIME 3,0,irpulse
  IF irpulse > 300 THEN remotecode.BIT3=1
 RCTIME 3,0,irpulse
  IF irpulse > 300 THEN remotecode.BIT4=1
 RCTIME 3,0,irpulse
  IF irpulse > 300 THEN remotecode.BIT5=1
 RCTIME 3,0,irpulse
  IF irpulse > 300 THEN remotecode.BIT6=1

RETURN                          'from get_ir_remotecode
RETURN                          'from TV-remote_control

'-------------------------------------------------------------
'Morse code generating subroutine
```

```
send_CW_ID:


  DIR6 = 1                'set pin 6 to output
  HIGH  6                 'set pin 6 high to signal 12F675 that
                        'Morse code is being sent in response
                        'to its trigger
'put your call sign here in dits and dahs


'W
  GOSUB dit
  GOSUB dah
  GOSUB dah
  PAUSE bit_space
'A
  GOSUB dit
  GOSUB dah
  PAUSE bit_space
'8
  GOSUB dah
  GOSUB dah
  GOSUB dah
  GOSUB dit
  GOSUB dit
  PAUSE bit_space
'S
```

```
  GOSUB dit
  GOSUB dit
  GOSUB dit
  PAUSE bit_space
'M
  GOSUB dah
  GOSUB dah
  PAUSE bit_space
'E
  GOSUB dit



  LOW 6                 'set pin 6 low to signal that Morse
                        'is complete
  PAUSE 10               'pause to make sure PIC sees complete
  DIR6 = 0              'set pin 6 to input
RETURN  'send_CW_ID



'-------------------------------------------------------------------
dit:
  FREQOUT spkr,dit_time,tone
  PAUSE dit_time
RETURN  'dit
```

```
dah:
  FREQOUT spkr,dah_time,tone
  PAUSE dit_time


RETURN  'dah


'--------------------------------------------------------------------
'read temperature subroutine

get_temp:
PAUSE 1000
HIGH rst
SHIFTOUT dq,clk,LSBFIRST,[rtemp]
SHIFTIN dq,clk,LSBPRE,[dsdata\9]
LOW rst
t_sign=sign
dsdata=dsdata/2
IF t_sign=0 THEN no_neg1
dsdata=dsdata|$ff00
no_neg1:

RETURN  'get_temp
```

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


'--------------------------------------------------------------------
get_light:

HIGH 10
PAUSE 100
RCTIME 10,1,light
light=65535/light*/647

light = NCD light       'conver light reading to log scale


RETURN  'get light


'get_compass subroutine

get_compass:
'DO                              ' Main loop
  PAUSE 100
  GOSUB Compass_Get_Axes                ' Get x, and y values
  GOSUB Compass_Correct_Offsets         ' Correct axis offsetes
  angle = x ATN -y                      ' Convert x and y to brads
```

```
  GOSUB Compass_Interpolate                ' Linear interpolation

  angle = angle */ 360                     ' Convert brads to degrees
  '
RETURN 'get_compass
' -----[ Subroutine - Compass_Get_Axes ]-------------------------------------

' This subroutine handles BASIC Stamp - HM55B communication and stores the
' magnetic field strength measurements returned by the device in the x and
' y axis variables.

Compass_Get_Axes:                          ' Compass module subroutine

  HIGH En:PAUSE 10: LOW En                 ' Send reset command to HM55B
  SHIFTOUT DinDout,clk,MSBFIRST,[Reset\4]

  HIGH En:PAUSE 10: LOW En                 ' HM55B start measurement command
  SHIFTOUT DinDout,clk,MSBFIRST,[Measure\4]
  status = 0                               ' Clear previous status flags

  DO                                       ' Status flag checking loop
    HIGH En: PAUSE 10: LOW En              ' Measurement status command
    SHIFTOUT DinDout,clk,MSBFIRST,[Report\4]
    SHIFTIN  DinDout,clk,MSBPOST,[Status\4]  ' Get Status
  LOOP UNTIL status = Ready                ' Exit loop when status is ready
```

```
  SHIFTIN  DinDout,clk,MSBPOST,[x\11,y\11]   ' Get x & y axis values
  HIGH En                                   ' Disable module

  IF (y.BIT10 = 1) THEN y = y | NegMask      ' Store 11-bits as signed word
  IF (x.BIT10 = 1) THEN x = x | NegMask      ' Repeat for other axis

  RETURN

' -----[ Subroutine - Compass_Correct_Offsets ]------------------------------

' This subroutine corrects cumulative magnetic field interference that can
' come from sources such as the PCB, jumper wires, a nearby battery, or a
' nearby current source.  This subroutine relies on values stored in
' the EEPROM space that was reserved by the CompassOffsets DATA directive.
' These EEPROM values were written by CalibrateHM55BCompass.bs2.

Compass_Correct_Offsets:

  READ CompassOffsets, Word axisOffset          ' Get x-axis offset
  x = x - axisOffset                            ' Correct x-axis
  READ CompassOffsets + 2, Word axisOffset      ' Get y-axis offset
  y = y - axisOffset                            ' Correct y-axis

  RETURN
```

```
' -----[ Subroutine - Compass_Interpolate ]----------------------------------

' This subroutine applies linear interpolation to the refine the compass
' measurement.  This second level of refinement can be performed after the
' Compass_Correct_Offsets subroutine, and it can correct axis skew and other
' errors inherent to the HM55B chip.
'
' The subroutine relies on sixteen actual compass measurements that were stored
' in the sixteen EEPROM locations reserved by the CompassCal DATA directive.
' These measurements were stored by CalibrateHM55BCompass.bs2, and they
' represent the actual compass measurements for 0, 22.5, 45, 90,..., 337.5
' degrees.  The subroutine finds the two EEPROM measurements that the current
' angle measurement falls between.  It then updates the angle measurement
' based on where the angle measurement falls between the two known table values.

Compass_Interpolate:

  ' Start with the lowest value in the CompassCal table.

  READ CompassLowVal, index

  ' Load current and previous table values.

  READ CompassCal + index, table(current)
```

```
READ (CompassCal + (index - 1 & $F)), table(previous)



' The IF...ELSEIF...ELSE...ENDIF code block finds the two EEPROM CompassCal
' table values that the current angle measurement falls between and calculates
' the difference between the current angle measurement and the lower of the
' two table values.  The IF and ELSEIF blocks deal with values that are
' greater than the highest or less than the lowest table values.  The ELSE
' block everything between the highest and lowest table values.

IF (angle >= table(previous)) THEN
  span = (255 - table(previous)) + table(current)
  angleOffset = angle - table(previous)
ELSEIF (angle <= table(current)) THEN
  span = table(current) + (255 - table(previous))
  angleOffset = angle + (255 - table(previous))
ELSE
  index = index - 1
  READ CompassCal + index, table(current)
  DO
    table(previous) = table(current)
    index = index + 1
    READ CompassCal + index, table(current)
    IF (angle <= table(current)) AND (angle > table(previous)) THEN
      span = table(current) - table(previous)
```

```
      angleOffset = angle - table(previous)
      EXIT
    ENDIF
  LOOP
ENDIF

' After the offset between the current angle measurement and the next lower
' table measurement has been determined, this code block uses it along with
' the span between the table entries above and below the angle measurement
' to solve for: angle(corrected) = angle(offset) * 16 / span.
' This code block also rounds up or down by comparing the remainder of
' the angleOffset / span division to the value of (span / 2).

angleOffset = angleOffset * 16
angle = (angleOffset / span) + ((angleOffset // span) / (span / 2))
angle = ((index - 1 & $F) * 16) + angle
angle = angle & $ff

RETURN 'get compass



'-------------------------------------------------------------------------
send_data:
```

```
  IF ( ABS distance <> distance)THEN
     distance_sign=0
     distance=65535-distance+1        'convert to positive number
                                      'for transmission
  ELSE
     distance_sign=2
  ENDIF
 GOSUB get_light
 GOSUB get_temp
'the UUUU is set to get the receiver in sync with the transmitter in a high
'RF environment because of the ATV TX on the BOT.
 SEROUT tpin,baudmode,50,["UUUU",CR,"DATA,",DEC angle,",",DEC distance_sign,",",
       DEC distance,",",DEC light, ",",DEC dsdata,CR]



 forwardflag=0
 distance=0

RETURN 'send_data
```