

BASIC listing: JOYSTB.BAS

```
*****
' *                               J O Y S T B                               * '
' * -----* '
' * Task           : Demonstrates joystick reading through BIOS.      * '
' * Author        : Michael Tischer                                  * '
' * Developed on   : 02/25/91                                          * '
' * Last update    : 04/07/95                                          * '
*****
DECLARE SUB GetJoyButton (j1b1%, j1b2%, j2b1%, j2b2%)
DECLARE SUB GetJoyPos (js1 AS ANY, js2 AS ANY)

REM $INCLUDE: 'g:qb.bi'

'-- Variable and type declarations -----

TYPE JSPOS 'Describes joystick position
  x AS INTEGER
  y AS INTEGER
END TYPE

DIM jsp(1 TO 2) AS JSPOS           'Current joystick position
DIM maxx AS INTEGER, maxy AS INTEGER 'Maximum joystick position
DIM minx AS INTEGER, miny AS INTEGER 'Minimum joystick position
DIM xold AS INTEGER, yold AS INTEGER 'Last joystick position
DIM curstick AS INTEGER           'Active joystick (1 or 2)
DIM xfactor AS SINGLE, yfactor AS SINGLE 'Coordinate factors
DIM j1but(1 TO 2) AS INTEGER      'Button 1 of joysticks 1 and 2
DIM j2but(1 TO 2) AS INTEGER      'Button 2 of joysticks 1 and 2

'-- Get maximum joystick positioning -----
```

```

CLS
PRINT "JOYSTICK POSITION TEST"
PRINT : PRINT "Push the joystick to the upper right, "
PRINT "then press one of the two buttons."

DO                                'Wait for a joystick button
    CALL GetJoyButton(jlbut(1), j2but(1), jlbut(2), j2but(2))
LOOP WHILE (jlbut(1) OR j2but(1) OR jlbut(2) OR j2but(2)) = 0

IF jlbut(1) OR j2but(1) <> 0 THEN    'Which joystick was that?
    curstick = 1
ELSE
    curstick = 2
END IF

CALL GetJoyPos(jsp(1), jsp(2))      'Read position
maxx = jsp(curstick).x              'Set as maximum position
miny = jsp(curstick).y

DO                                'Wait for release of a joystick button
    CALL GetJoyButton(jlbut(1), j2but(1), jlbut(2), j2but(2))
LOOP UNTIL (jlbut(curstick) = 0) AND (j2but(curstick) = 0)

'-- Get minimum joystick positioning -----

PRINT : PRINT
PRINT "Push the joystick to the lower left, "
PRINT "then press one of the two buttons."

DO                                'Wait for a joystick button
    CALL GetJoyButton(jlbut(1), j2but(1), jlbut(2), j2but(2))
LOOP WHILE (jlbut(curstick) = 0) AND (j2but(curstick) = 0)

```

```

CALL GetJoyPos(jsp(1), jsp(2))                                'Read position
minx = jsp(curstick).x                                        'Set as minimum position
maxy = jsp(curstick).y

factorx = 80 / (maxx - minx + 1)                              'Compute coordinate factor
factory = 23 / (maxy - miny + 1)                              'using X-axis and Y-axis

'-- Read joystick, display position until -----
'-- the user presses both joystick buttons -----

CLS
LOCATE 2, 44
PRINT "JOYSTB - (c) 1992 by Michael Tischer";
LOCATE 25, 1
PRINT "Press both joystick buttons to end the program"

xold = 1                                                       'Set old position
yold = 1

DO
  '-- Read and display position -----
  CALL GetJoyPos(jsp(1), jsp(2))
  LOCATE 1, 1
  PRINT "("; jsp(curstick).x; "/" ; jsp(curstick).y; ")"  ";

  '-- Compute new X-position of the joystick -----
  x% = factorx * (jsp(curstick).x - minx% + 1)
  IF x% < 1 THEN x% = 1
  IF x% > 80 THEN x% = 80

  '-- Compute new Y-position of the joystick -----

```

```

y% = factory * (jsp(curstick).y - miny% + 1)
IF y% < 1 THEN y% = 1
IF y% > 23 THEN y% = 23

'-- Display new position if position changes -----
IF (x% <> xold) OR (y% <> yold) THEN
    LOCATE yold + 1, xold                                'Mask at old X-position
    PRINT " ";
    LOCATE y% + 1, x%                                     'and display at new X-position
    PRINT "X";
    xold = x%                                             'Make new position old position
    yold = y%
END IF

CALL GetJoyButton(j1but(1), j2but(1), j1but(2), j2but(2))
LOOP UNTIL (j1but(curstick) = 1) AND (j2but(curstick) = 1)

CLS

' *****
' * GetJoyButton: Returns joystick button status. *
' *-----*
' * Input   : J1B1% = 1 if button 1 (stick 1) depressed, otherwise 0 *
' *          J1B2% = 1 if button 2 (stick 1) depressed, otherwise 0 *
' *          J2B1% = 1 if button 1 (stick 2) depressed, otherwise 0 *
' *          J2B2% = 1 if button 2 (stick 2) depressed, otherwise 0 *
' *****
'

SUB GetJoyButton (j1b1%, j1b2%, j2b1%, j2b2%)
    DIM regs AS RegType                                'Dimension array for registers

    regs.ax = &H8400                                    'Call BIOS interrupt 15H, function 84H,

```

```

regs.dx = &H0                                'sub-function 00H
CALL INTERRUPT(&H15, regs, regs)
j1b1% = (regs.ax AND 16) \ 16 XOR 1           'Bit 4 of AX = J1B1
j1b2% = (regs.ax AND 32) \ 32 XOR 1           'Bit 5 of AX = J1B2
j2b1% = (regs.ax AND 64) \ 64 XOR 1           'Bit 6 of AX = J2B1
j2b2% = (regs.ax AND 128) \ 128 XOR 1         'Bit 7 of AX = J2B2
END SUB

```

```

' ***** '
' * GetJoyPos : Gets positions of both joysticks. * '
' * ----- * '
' * Input : JS1 = Structure of type JSPOS for storing position of * '
' *          joystick 1 * '
' *          JS2 = Structure of type JSPOS for storing position of * '
' *          joystick 2 * '
' ***** '
'

```

```

SUB GetJoyPos (js1 AS JSPOS, js2 AS JSPOS)
  DIM regs AS RegType 'Dimension array for registers

  regs.ax = &H8400      'Call BIOS interrupt 15H, function 84H,
  regs.dx = &H1         'sub-function 01H
  CALL INTERRUPT(&H15, regs, regs)
  js1.x = regs.ax        'Store joystick position registers
  js1.y = regs.bx        'in respective variables
  js2.x = regs.cx
  js2.y = regs.dx
END SUB

```