

C listing: JOYSTC.C

```

/*****
/*
/*          J O Y S T C
/*
/*-----*/
/*   Task           : Demonstrates joystick reading through BIOS.   */
/*-----*/
/*   Author          : Michael Tischer                               */
/*   Developed on    : 02/25/91                                       */
/*   Last update     : 04/07/95                                       */
/*-----*/
/*   (MICROSOFT C)
/*   Compilation     : CL /AS JOYSTC.C                               */
/*   Call            : JOYSTC                                         */
/*-----*/
/*   (BORLAND TURBO C)
/*   Compilation     : Use COMPILE/MAKE command                      */
*****/
```

```
/*== Add include files =====*/
```

```
#include <dos.h>
#include <stdio.h>
#include <stdarg.h>
```

```
/*== Type declarations =====*/
```

```
typedef unsigned char BYTE;
typedef struct {
    int x;
    int y;
} JSPOS;
/* Describes joystick position */
```

```

/*****
*   ClrScr: Clears the screen.
*   -----
*   Input : DCOLR = Character attribute
*   Output: None
*****/

```

```
void ClrScr( BYTE dcolr )
```

```

{
    union REGS regs;          /* Processor regs. for interrupt call */

    /*-- Clear screen using BIOS scroll function -----*/

    regs.h.ah = 6;             /* Function number: Scroll down */
    regs.h.al = 0;             /* Scroll 0 lines (clear screen) */
    regs.h.bh = dcolr;         /* Character color */
    regs.x.cx = 0;             /* Upper-left corner of window */
    regs.x.dx = ( 24 << 8 ) + 79; /* Lower-right corner of window */
    int86(0x10, &regs, &regs); /* Call BIOS video interrupt */

    /*-- Place cursor in upper-left corner using BIOS -----*/

    regs.h.ah = 2;             /* Function number: Set cursor */
    regs.h.bh = 0;             /* Access screen page 0 */
    regs.x.dx = 0;             /* Upper-left corner of screen */
    int86(0x10, &regs, &regs); /* Call BIOS video interrupt */
}

```

```

/*****
*   printfat : Displays a formatted string anywhere on the screen.
*   -----
*   Input : COLUMN = Column
*

```

```

*          SCROW  = Row                      *
*          STRING = Pointer to string        *
*          ...    = Arguments similar to PRINTF() *
*   Output: None                             *
*   Info  : This function should only be called if the system running *
*          this program contains an EGA card or a VGA card.          *
*****/

```

```

void printfat( BYTE column, BYTE scrow, char * string, ... )
{
    va_list parameter;                      /* Parameter list for VA... macros */
    union REGS regs;                        /* Processor regs. for interrupt call */

    /*-- Place cursor using BIOS -----*/

    regs.h.ah = 2;                          /* Function number: Set cursor */
    regs.h.bh = 0;                          /* Access screen page 0 */
    regs.h.dh = scrow;                      /* Set row */
    regs.h.dl = column;                    /* Set column */
    int86(0x10, &regs, &regs);             /* Call BIOS video interrupt */

    /*-- Display string -----*/

    va_start( parameter, string );
    vprintf( string, parameter );
}

```

```

/*****
*   Function      : G E T J O Y B U T T O N *
**-----**
*   Task   : Returns joystick button status. *
*   Input  : J1B1 = 1 if button 1 (stick 1) depressed, otherwise 0 *

```



```

regs.h.ah = 0x84;                                /* Function 84H */
regs.x.dx = 1;                                    /* Sub-function 01H */
int86( 0x15, &regs, &regs );                     /* Call interrupt 15H */
Js1Ptr->x = regs.x.ax;                             /* X-position: Joystick 1 */
Js1Ptr->y = regs.x.bx;                             /* Y-position: Joystick 1 */
Js2Ptr->x = regs.x.cx;                             /* X-position: Joystick 2 */
Js2Ptr->y = regs.x.dx;                             /* Y-position: Joystick 2 */
}

```

```

/*****
**                                     **
MAIN PROGRAM                                     **
*****/

```

```

void main()
{
    JSPOS jsp[2];                                /* Current joystick position */
    int maxx, maxy,                             /* Maximum joystick position */
        minx, miny,                             /* Minimum joystick position */
        x, y,                                    /* Current screen position */
        xold, yold;                             /* Last screen position */
    BYTE curstick,                               /* Active joystick */
        j1but[2],                               /* Button 1 of joysticks 1 and 2 */
        j2but[2];                               /* Button 2 of joysticks 1 and 2 */
    float xfactor, yfactor;                     /* Coordinate factors X and Y */

    /*-- Get maximum joystick positioning -----*/

    ClrScr( 0x07 );
    printf( "JOYSTICK POSITION TEST\n\n");
    printf( "Push the joystick to the upper right,\n" \
        "then press one of the two buttons.\n" );
}

```

```

do
                                /* Wait for a joystick button */
    GetJoyButton( &jlbut[0], &j2but[0], &jlbut[1], &j2but[1] );
    while ( ( jlbut[0] | j2but[0] | jlbut[1] | j2but[1] ) == 0 );

    curstick = ( jlbut[0] | j2but[0] ) ? 0 : 1;          /* Select joystick */

    GetJoyPos( &jsp[0], &jsp[1] );                      /* Read position */
    maxx = jsp[curstick].x;                             /* Set position */
    miny = jsp[curstick].y;

do
                                /* Wait for release of a joystick button */
    GetJoyButton( &jlbut[0], &j2but[0], &jlbut[1], &j2but[1] );
    while ( ( jlbut[curstick] | j2but[curstick] ) != 0 );

    /*-- Get minimum joystick positioning -----*/

    printf( "\n\nPush the joystick to the lower left,\n\n\
            "then press one of the two buttons.\n" );

do
                                /* Wait for a joystick button */
    GetJoyButton( &jlbut[0], &j2but[0], &jlbut[1], &j2but[1] );
    while ( ( jlbut[curstick] | j2but[curstick] ) == 0 );

    GetJoyPos( &jsp[0], &jsp[1] );                      /* Read position */
    minx = jsp[curstick].x;                             /* Set position */
    maxy = jsp[curstick].y;

do
                                /* Wait for release of a joystick button */
    GetJoyButton( &jlbut[0], &j2but[0], &jlbut[1], &j2but[1] );
    while ( ( jlbut[curstick] | j2but[curstick] ) != 0 );

    xfactor = 80.0 / ( maxx - minx + 1 ); /* Compute coordinate factor */

```

```

yfactor = 23.0 / ( maxy - miny + 1 ); /* using X-axis and Y-axis */

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

ClrScr( 0x07 );
printfat( 43, 0, "JOYSTC - (c) 1992 by Michael Tischer" );
printfat( 0, 24, "Press both joystick buttons to end the program" );

xold = yold = 0; /* Set old position */
do
{
    GetJoyPos( &jsp[0], &jsp[1] ); /* Read position */

    /*-- Compute new X-position of the joystick -----*/

    x = (int) ( xfactor * (float) ( jsp[curstick].x - minx + 1 ) );
    if ( x < 0 )
        x = 0;
    if ( x > 79 )
        x = 79;

    /*-- Compute new X-position of the joystick -----*/

    y = (int) ( yfactor * (float) ( jsp[curstick].y - miny + 1 ) );
    if ( y < 0 )
        y = 0;
    if ( y > 22 )
        y = 22;

    /*-- Display new position if position changes -----*/

```

```
if ( x != xold || y != yold )
{
    printfat( (BYTE) xold, (BYTE) (yold+1), " " );
    printfat( (BYTE) x, (BYTE) (y+1), "X" );
    xold = x;
    yold = y;
}

printfat( 0, 0, "(%3d,%3d)", jsp[curstick].x, jsp[curstick].y );
GetJoyButton( &jlbut[0], &j2but[0], &jlbut[1], &j2but[1] );
}
while ( jlbut[curstick] == 0 && j2but[curstick] == 0 );
ClrScr( 0x07 );
printf( "End program.\n");
```