DirectInput

DirectInput

DirectInput

Introduction to Joysticks

The Microsoft® DirectInput™ application programming interface (API) provides fast and consistent access to analog and digital joysticks. The DirectInput API maintains consistency with the joystick APIs of the Microsoft Win32® Software Development Kit (SDK), but has improved responsiveness and reliability by changing the device driver model. DirectInput device drivers also use the registry to store settings for standard joysticks, calibration information for previously configured joysticks, and settings for OEM-supplied joysticks.

This topic describes DirectInput functions, messages, and structures that support joysticks and serves as an update to the Win32 joystick API. The DirectInput APIs also apply to other ancillary input devices that track positions within an absolute coordinate system, such as a touch screen, digitizing tablet, and light pen. Extended capabilities also provide support for rudder pedals, flight yokes, virtual-reality headgear, and other devices. Each device can use up to six axes of movement, a point-of-view hat, and 32 buttons.

You can use DirectInput functions to determine the capabilities of the joysticks and joystick drivers. Also, you can process a joystick's position and button information by querying the joystick.

Joystick Capabilities

DirectInput services are loaded when the operating system is started. DirectInput supports analog and digital joysticks. Analog joysticks require real-time responsiveness and place a higher burden on the system than digital joysticks. DirectInput can simultaneously monitor two analog joysticks that track up to four axes of movement and use up to four buttons. Alternatively, the services can simultaneously monitor up to four analog joysticks that track two axes of movement and use up to four buttons. DirectInput can support up to 16 digital joysticks, each with up to six axes of movement and up to 32 buttons.

Each axis of movement that a joystick tracks has a *range of motion*. The range of motion is the distance a joystick handle can move from its neutral (resting) position to the position farthest from its resting position.

The joystick driver can support up to 16 minidrivers. Each minidriver supports one joystick. You can retrieve the number of joysticks supported by a joystick driver by using the **joyGetNumDevs** function. This function returns an unsigned integer specifying the number of joysticks that the driver can support or zero if there is no joystick support.

You can determine if a joystick is attached to the system by using the **joyGetPosEx** function. This function returns JOYERR_NOERROR if the specified device is attached or JOYERR_UNPLUGGED otherwise.

Each joystick has several capabilities that are available to your application. You can retrieve the capabilities of a joystick by using the **joyGetDevCaps** function. This function fills a **JOYCAPS** structure with joystick capabilities such as the valid axes of movement for the joystick, minimum and maximum values for its coordinate system, and the number of buttons on the joystick.

Note

The return value of **joyGetNumDevs** does not indicate the number of joysticks attached to the system, rather the number of joysticks that the system supports.

Joystick Calibration and Testing

Microsoft Windows® 95 provides a joystick application in the Control Panel to calibrate and test joysticks, their ranges of motion and buttons: the application lets the user choose from a list of joysticks, including the following:

- Generic joystick configurations
- OEM joysticks
- Custom joysticks

The application allows the calibration of up to six axes of motion, 32 buttons, and a point-of-view hat for each joystick. Calibration information is stored in the registry, so a user can switch between joysticks without the need to recalibrate a joystick each time it's used. Once the user has calibrated or selected a new joystick, the calibration application updates the registry with the selected joystick and calibration information, and notifies the joystick driver accordingly.

Additionally, your application can notify a joystick of changes to the registry that affect the joystick by using the **joyConfigChanged** function.

Joystick Position

You can query a joystick for position and button information by using the **joyGetPosEx** function. This function reports position information returned by the older joystick functions of the Win32 API., including position coordinates for the x-, y-, and z- axes, and state information for up to four buttons. The **joyGetPosEx** function also provides access to the following information:

- Fourth, fifth, and sixth axes (r, u, and v)
- Rudder information
- Point-of-view hat
- State information for up to 32 buttons
- Uncalibrated (raw) joystick data
- Scaled data for a defined range of values
- Centered scaled data
- Scaled data that includes a dead zone around the neutral joystick position

Joystick Groups

This topic describes the functions and structures associated with joysticks. The elements are grouped as follows:

- Device Capabilities
- Querying a Joystick

Joystick Groups

This topic describes the functions and structures associated with joysticks. The elements are grouped as follows:

- Device Capabilities
    - joyGetDevCaps
    - joyGetNumDevs
    - joyConfigChanged
    - JOYCAPS
- Querying a Joystick

Joystick Groups

This topic describes the functions and structures associated with joysticks. The elements are grouped as follows:

Functions

Functions

An application uses the joystick functions to query a joystick driver and to notify the driver of registry changes affecting a joystick.

joyConfigChanged

```
MMRESULT joyConfigChanged(DWORD dwFlags);
```

Notifies the joystick driver that the registry contains new joystick settings.

- Returns JOYERR_NOERROR if successful or one of the following error values:

JOYERR_REGISTRYNOTVALID	One or more registry joystick entries contain invalid data.
JOYERR_NOCANDO	The joystick driver cannot update the device information from the registry.

dwFlags

Reserverd. Must be zero.

The joystick calibration property sheet in the control panel calls this function when the user calibrates or recalibrates a joystick, or when the user selects a different joystick device.

Applications that customize joystick performance, such as OEM joystick calibration applications, can use this function to notify the joystick driver that the JOYSTICK_USER values in the registry for the currently selected joystick have changed. The JOYSTICK_USER values are located in the HKEY_LOCAL_MACHINE hive of the registry.

joyGetDevCaps

```
MMRESULT joyGetDevCaps(UINT uJoyID, LPJOYCAPS pjc, UINT cbjc);
```

Queries a joystick to determine its capabilities.

- Returns JOYERR_NOERROR if successful or one of the following error values:

MMSYSERR_INVALPARAM	An invalid parameter was passed.
MMSYSERR_NODRIVER	The joystick driver is not present.

uJoyID
Identifier of the joystick (JOYSTICKID1 or JOYSTICKID2) to be queried.

pjc
Address of a **JOYCAPS** structure to contain the capabilities of the joystick.

cbjc
Size, in bytes, of the **JOYCAPS** structure.

Use the **joyGetNumDevs** function to determine the number of joystick devices supported by the driver.

joyGetNumDevs

```
UINT joyGetNumDevs(VOID);
```

Queries the joystick driver for the number of joysticks it supports.

- Returns the number of joysticks supported by the joystick driver or zero if no driver is present.

Use the **joyGetPos** function to determine whether a given joystick is physically attached to the system.

joyGetPosEx

```
MMRESULT joyGetPosEx(UINT uJoyID, LPJOYINFOEX pji);
```

Queries a joystick for its position and button status.

- Returns JOYERR_NOERROR if successful or one of the following error values:

JOYERR_UNPLUGGED	The specified joystick is not connected to the system.
MMSYSERR_BADDEVICEID	The specified joystick identifier is invalid.
MMSYSERR_INVALPARAM	An invalid parameter was passed.
MMSYSERR_NODRIVER	The joystick driver is not present.

uJoyID

Identifier of the joystick to be queried.

pji

Address of a **JOYINFOEX** structure that contains extended position information and button status of the joystick.

Before calling this function, an application must identify the items to query by setting one or more flags in the **dwFlags** member of **JOYINFOEX**.

This function provides access to extended devices such as rudder pedals, point-of-view hats, devices with a large number of buttons, and coordinate systems using up to six axes.

Structures

Structures

The following structures store information about the capabilities of a joystick or its current position and button states.

JOYCAPS

```
typedef struct {
    WORD wMid;                 \\ manufacturer identifier
    WORD wPid;                 \\ product identifier
    CHAR szPname[MAXPNAMELEN];  \\ see below
    UINT wXmin;                \\ min. x-coordinate
    UINT wXmax;                \\ max. x-coordinate
    UINT wYmin;                \\ min. y-coordinate
    UINT wYmax;                \\ max. y-coordinate
    UINT wZmin;                \\ min. z-coordinate
    UINT wZmax;                \\ max. z-coordinate
    UINT wNumButtons;          \\ no. of joystick buttons
    UINT wPeriodMin;           \\ see below
    UINT wPeriodMax;           \\ see below
\\  The following members are not in previous versions of Windows.
    UINT wRmin;                \\ see below
    UINT wRmax;                \\ see below
    UINT wUmin;                \\ see below
    UINT wUmax;                \\ see below
    UINT wVmin;                \\ see below
    UINT wVmax;                \\ see below
    UINT wCaps;                \\ see below
    UINT wMaxAxes;             \\ see below
    UINT wNumAxes;             \\ see below
    UINT wMaxButtons;          \\ see below
    CHAR szRegKey[MAXPNAMELEN]; \\ see below
    CHAR szOEMVxD[MAXOEMVXD];   \\ see below
} JOYCAPS;
```

Contains information about the joystick capabilities.

szPname
Null-terminated string containing the joystick product name.

wPeriodMin
Smallest polling frequency supported when an application has captured a joystick.

wPeriodMax
Largest polling frequency supported an application has captured a joystick.

wRmin and **wRmax**
Minimum and maximum rudder values. The rudder is a fourth axis of movement.

wUmin and **wUmax**
Minimum and maximum u-coordinate (fifth axis) values.

wVmin and **wVmax**
Minimum and maximum v-coordinate (sixth axis) values.

wCaps
Joystick capabilities The following flags define individual capabilities that a joystick might have:

JOYCAPS_HASPOV	Joystick has point-of-view information.
JOYCAPS_HASR	Joystick has rudder (fourth axis) information.
JOYCAPS_HASU	Joystick has u-coordinate (fifth axis) information.
JOYCAPS_HASV	Joystick has v-coordinate (sixth axis) information.
JOYCAPS_HASZ	Joystick has z-coordinate information.
JOYCAPS_POV4DIR	Joystick point-of-view supports discrete values (centered, forward, backward, left, and right).
JOYCAPS_POVCTS	Joystick point-of-view supports continuous degree bearings.

wMaxAxes

Maximum number of axes supported by the joystick.

wNumAxes

Number of axes currently in use by the joystick.

wMaxButtons

Maximum number of buttons supported by the joystick.

szRegKey

Null-terminated string containing the registry key for the joystick.

szOEMVxD

Null-terminated string identifying the joystick driver OEM.

JOYINFOEX

```
typedef struct joyinfoex_tag {
    DWORD dwSize;         \\ size, in bytes, of this structure
    DWORD dwFlags;        \\ see below
    DWORD dwXpos;         \\ current x-coordinate
    DWORD dwYpos;         \\ current y-coordinate
    DWORD dwZpos;         \\ current z-coordinate
    DWORD dwRpos;         \\ see below
    DWORD dwUpos;         \\ current 5th axis position
    DWORD dwVpos;         \\ current 6th axis position
    DWORD dwButtons;      \\ see below
    DWORD dwButtonNumber; \\ see below
    DWORD dwPOV;          \\ see below
    DWORD dwReserved1;    \\ reserved; do not use
    DWORD dwReserved2;    \\ reserved; do not use
} JOYINFOEX;
```

Contains extended information about the joystick position, point-of-view position, and button state.

dwFlags

Flags indicating the valid information returned in this structure. Members that do not contain valid information are set to zero. The following flags are defined:

JOY_RETURNALL	Equivalent to setting all of the JOY_RETURN bits except JOY_RETURNRAWDATA.
JOY_RETURNBUTTONS	The **dwButtons** member contains valid information about the state of each joystick button.
JOY_RETURNCENTERED	Centers the joystick neutral position to the middle value of each axis of movement.
JOY_RETURNPOV	The **dwPOV** member contains valid information about the point-of-view control, expressed in discrete units.
JOY_RETURNPOVCTS	The **dwPOV** member contains valid information about the point-of-view control expressed in continuous, one-hundredth degree units.
JOY_RETURNR	The **dwRpos** member contains valid rudder pedal data. This information represents another (fourth) axis.
JOY_RETURNRAWDATA	Data stored in this structure is uncalibrated joystick readings.
JOY_RETURNU	The **dwUpos** member contains valid data for a fifth axis of the joystick, if such an axis is available, or returns zero otherwise.
JOY_RETURNV	The **dwVpos** member contains valid data for a sixth axis of the joystick, if such an axis is available, or returns zero otherwise.
JOY_RETURNX	The **dwXpos** member contains valid data for the x-coordinate of the joystick.
JOY_RETURNY	The **dwYpos** member contains valid data for the y-coordinate of the joystick.
JOY_RETURNZ	The **dwZpos** member contains valid data for the z-coordinate of the joystick.

JOY_USEDEADZONE	Expands the range for the neutral position of the joystick and calls this range the dead zone. The joystick driver returns a constant value for all positions in the dead zone.

The following flags provide data to calibrate a joystick and are intended for custom calibration applications:

JOY_CAL_READ3	Read the x-, y-, and z-coordinates and store the raw values in **dwXpos**, **dwYpos**, and **dwZpos**.
JOY_CAL_READ4	Read the rudder information and the x-, y-, and z-coordinates and store the raw values in **dwXpos**, **dwYpos**, **dwZpos**, and **dwRpos**.
JOY_CAL_READ5	Read the rudder information and the x-, y-, z-, and u-coordinates and store the raw values in **dwXpos**, **dwYpos**, **dwZpos**, **dwRpos**, and **dwUpos**.
JOY_CAL_READ6	Read the raw v-axis data if a joystick minidriver is present that will provide the data. Return zero otherwise.
JOY_CAL_READALWAYS	Read the joystick port even if the driver does not detect a device.
JOY_CAL_READRONLY	Read the rudder information if a joystick minidriver is present that will provide the data and store the raw value in **dwRpos**. Return zero otherwise.
JOY_CAL_READUONLY	Read the u-coordinate if a joystick minidriver is present that will provide the data and store the raw value in **dwUpos**. Return zero otherwise.
JOY_CAL_READVONLY	Read the v-coordinate if a joystick minidriver is present that will provide the data and store the raw value in **dwVpos**. Return zero otherwise.
JOY_CAL_READXONLY	Read the x-coordinate and store the raw (uncalibrated) value in **dwXpos**.
JOY_CAL_READXYONLY	Reads the x- and y-coordinates and place the raw values in **dwXpos** and **dwYpos**.
JOY_CAL_READYONLY	Reads the y-coordinate and store the raw value in **dwYpos**.
JOY_CAL_READZONLY	Read the z-coordinate and store the raw value in **dwZpos**.

dwRpos

Current position of the rudder or fourth joystick axis.

dwButtons

Current state of the 32 joystick buttons. The value of this member can be set to any combination of JOY_BUTTON*n* flags, where *n* is a value in the range of 1 through 32 corresponding to the button that is pressed.

dwButtonNumber

Current button number that is pressed.

dwPOV

Current position of the point-of-view control. Values for this member are in the range 0 through 35,900. These values represent the angle, in degrees, of each view multiplied by 100.

The value of the **dwSize** member is also used to identify the version number for the structure when it’s passed to the **joyGetPosEx** function.

Most devices with a point-of-view control have only five positions. When the JOY_RETURNPOV flag is set, these positions are reported by using the following constants.

Point-of-view flags	**Description**
JOY_POVBACKWARD	Point-of-view hat is pressed backward. The value 18,000 represents an orientation of 180.00 degrees (to the rear).
JOY_POVCENTERED	Point-of-view hat is in the neutral position. The value -1 means the point-of-view hat has no angle to report.
JOY_POVFORWARD	Point-of-view hat is pressed forward. The value 0 represents an orientation of 0.00 degrees (straight ahead).
JOY_POVLEFT	Point-of-view hat is being pressed to the left. The value 27,000 represents an orientation of 270.00 degrees (90.00 degrees to the left).
JOY_POVRIGHT	Point-of-view hat is pressed to the right. The value 9,000 represents an orientation of 90.00 degrees (to the right).

The default Windows 95 joystick driver currently supports these five discrete directions. If an application can accept only the defined point-of-view values, it must use the JOY_RETURNPOV flag. If an application can accept other degree readings, it should use the JOY_RETURNPOVCTS flag to obtain continuous data if it is available. The JOY_RETURNPOVCTS flag also supports the JOY_POV constants used with the JOY_RETURNPOV flag.