12

Human Interface Devices: Reports

Chapter 11 introduced the reports that HIDs use to exchange data. A report can be a basic buffer of bytes or a complex assortment of items, each with assigned functions and units. This chapter shows how to design a report to fit a specific application.

Report Structure

A report descriptor may contain any of dozens of items arranged in various combinations. The advantage of a more complex descriptor is that the device can provide detailed information about the data it sends and expects to receive. The descriptor can specify the values' uses and what units to apply to the raw data. From a report descriptor, an application can learn whether a device supports a particular feature, such as force feedback on a joystick.

But just because the specification defines an item that could apply to a device's data doesn't mean that the report descriptor has to use that item. For vendor-specific devices that are intended for use with a single application, the application often knows in advance the type, size, and order of the data in a report, so there's no need to obtain this information from the device. For example, when the vendor of a data-acquisition unit creates an application for use with the unit, the vendor already knows the data format the device uses in its reports. At most, the application might check the product ID and release number from the device descriptor to learn whether the application can request a particular setting or action.

Some of the details about report structures can get tedious, and it's rarely necessary to understand every nuance about every item type. So feel free to skim through the details in this chapter and come back to them later if you need to.

A report descriptor contains one or more controls and data items that describe values to be transferred in one or more reports. A control is a button, switch, or other physical entity that operates or regulates an aspect of a device. Data items describe any other data the report transfers. Each control or data item has a defined scope. Some items can apply to multiple values, eliminating the need for repetition.

Using the HID Descriptor Tool

The HID Descriptor Tool (Figure 12-1) is a free utility available from the USB-IF. The tool helps in creating report descriptors, and will also check your descriptor's structure and report errors. Instead of having to look up the values that correspond to each item in your report, you can select the item from a list and enter the value you want to assign to it, and the tool adds the item to the descriptor. You can also add items manually. The Parse Descriptor function displays the raw and interpreted values in your descriptor and comments on any errors found. When you have a descriptor with no errors, you can convert it to the syntax required by your firmware. The tool has limited support for vendor-specific items, however, and may flag these as errors.

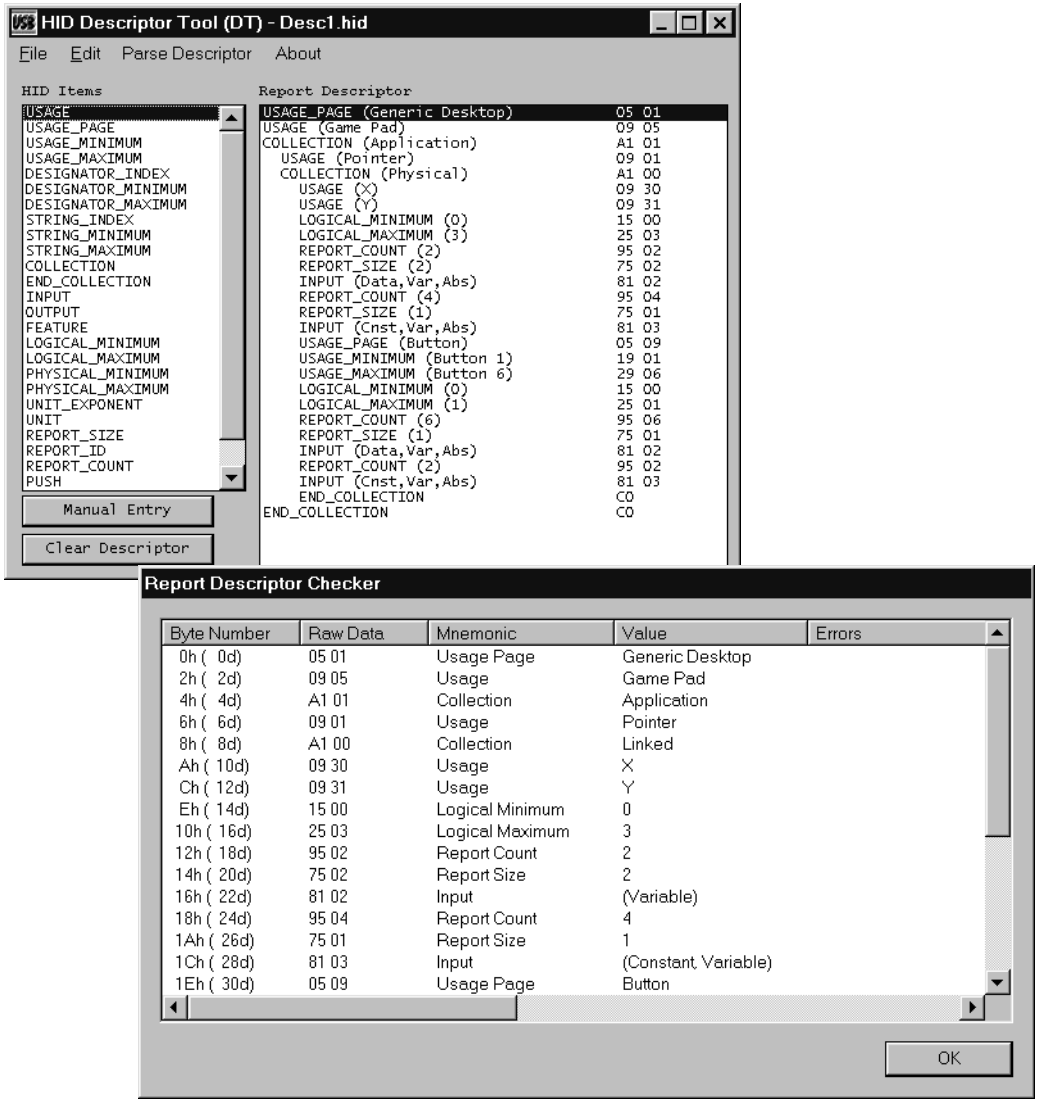

Figure 12-1: The HID Descriptor Tool helps in creating and testing HID report descriptors.

Control and Data Item Values

Several documents define values that reports may contain. The first place to look is the *HID Usage Tables* document, which defines values for generic desktop controls, simulation controls, game controls, LEDs, buttons, telephone devices, and more. The document also tells you where to find values defined elsewhere. Some are in the HID specification, while others are in the class specifications for specific device functions such as monitor, power, and point-of-sale devices.

Item Types

The HID specification defines two report item types: short items and long items. As of HID 1.1, there are no defined Long items, and the type is just reserved for future use.

Short Items

A Short item's 1-byte prefix specifies the item type, item tag, and item size. These are the elements that make up the prefix byte:

Bit Number	Contents	Description
0	Item Size	Number of bytes in the item
1		
2	Item Type	Item scope: Main, Global, or Local
3		
4	Item Tag	Numeric value that indicates the item's function
5		
6		
7		

The item size (bits 1 and 0) indicates how many data bytes the item contains. Note that an item size of 3 (11 in binary) corresponds to 4 data bytes:

Item Size (binary)	Number of Data Bytes
00	0
01	1
10	2
11	4

The item type (bits 3 and 2) describes the scope of the item: Main (00), Global (01), or Local (10). Main items define or group data fields in the descriptor. Global items describe data. Local items define characteristics of individual controls or data. (This chapter has more information about these item types.)

The item tag (bits 4-7) indicates the item's function.

Long Items

A Long item uses multiple bytes to store the same information as the Short item's 1-byte prefix stores. A Long item's 1-byte prefix (FEh) identifies the item as a Long item. In addition, the item has a byte that specifies the number of data bytes, a byte containing the item tag, and up to 255 bytes of data.

The Main Item Type

A Main item defines or groups data items within a report descriptor. There are five Main item types. Input, Output, and Feature items each define fields in a type of report. Collection and End Collection items group related items within a report. The default value for all Main items is zero.

Input, Output, and Feature Items

Table 12-1 shows supported values for the Input, Output, and Feature items, including the item prefix and the meanings of the bits in the data that follows the prefix.

An Input item applies to data a device sends to the host. An Input report contains one or more Input items. The host uses interrupt IN transfers or Get_Report requests to request Input reports.

An Output item applies to information that the host sends to the device. An Output report contains one or more Output items. Output reports contain data that reports the states of controls, such as whether to open or close a switch or the intensity to apply to an effect. As explained earlier, if the HID has an interrupt OUT endpoint, a HID 1.1-compliant host can use interrupt OUT transfers to send Output reports. All hosts can also use Set_Report requests to send Output reports.

A Feature item typically applies to information that the host sends to the device. However, it's also possible for the host to read Feature items from a device. A Feature report contains one or more Feature items. Feature reports often contain configuration settings that affect the overall behavior of the device or one of its components. The reports often control settings that you might otherwise adjust in a physical control panel. For example, the host may have a virtual (on-screen) control panel to enable users to select and control a device's capabilities and settings. The host uses control transfers with Set_Report and Get_Report requests to send and receive Feature reports.

Following each Input, Output, or Feature item prefix are up to 9 bits that describe the item's data. (An additional 23 bits are reserved.) For example, an Input item prefix followed by 9 bits of data has the value 82h. The prefix's high four bits equal 08h to indicate an Input item, and the low four bits equal 02h to indicate that the data's 9 bits require 2 bytes. An Input item prefix followed by 8 bits of data has the value 81h. The prefix's high four bits equal 08h to indicate an Input item, and the low four bits equal 01h to because the data's 8 bits fit in 1 byte. The device firmware and host software may use or ignore the information in these items.

Table 12-1: The bits that follow Input, Output, and Feature Item prefixes describe the data in report items.

Main Item Prefix	Bit Number	Meaning if bit = 0	Meaning if bit = 1
Input (100000nn, where nn=the number of bytes in the data following the prefix) Use 81h for 1 byte of item data or use 82h for 2 bytes of item data.	0	Data	Constant
	1	Array	Variable
	2	Absolute	Relative
	3	No wrap	Wrap
	4	Linear	Non-linear
	5	Preferred state	No preferred state
	6	No null position	Null state
	7	Reserved	
	8	Bit field	Buffered bytes
	9-31	Reserved	
Output (100100nn, where nn=the number of bytes in the data following the prefix) Use 91h for 1 byte of item data or use 92h for 2 bytes of item data.	0	Data	Constant
	1	Array	Variable
	2	Absolute	Relative
	3	No wrap	Wrap
	4	Linear	Non-linear
	5	Preferred state	No preferred state
	6	No null position	Null state
	7	Non-volatile	Volatile
	8	Bit field	Buffered bytes
	9-31	Reserved	
Feature (101100nn, where nn=the number of bytes in the data following the prefix) Use B1h for 1 byte of item data or use B2h for 2 bytes of item data.	0	Data	Constant
	1	Array	Variable
	2	Absolute	Relative
	3	No wrap	Wrap
	4	Linear	Non-linear
	5	Preferred state	No preferred state
	6	No null position	Null state
	7	Non-volatile	Volatile
	8	Bit field	Buffered bytes
	9-31	Reserved	

The bit functions are the same for Input, Output, and Feature items, except that Input items don't support the volatile/non-volatile bit. These are the uses for each bit:

Data | Constant. Data means that the contents of the item are modifiable (read/write). Constant means the contents are not modifiable (read-only).

Array | Variable. This bit specifies whether the data reports the state of every control (Variable) or just reports the states of controls that are asserted, or active (Array). Reporting only the asserted controls results in a more compact report for devices such as keyboards, where there are many controls (keys) but only one or a few are asserted at the same time.

For example, if a keypad has eight keys, setting this bit to Variable would mean that the keypad's report would contain a bit for each key. In the report descriptor, the report size would be one bit, the report count would be eight, and the total amount of data sent would be eight bits. Setting the bit to Array would mean that each key has an assigned index, and the keypad's report would contain only the indexes of keys that are pressed. With eight keys, the report size would be three bits, which can report a key number from 0 through 7. The report count would equal the maximum number of simultaneous keypresses that could be reported. If the user can press only one key at a time, the report count would be 1 and the total amount of data sent would be just 3 bits. If the user can press all of the keys at once, the report count would be 8 and the total amount of data sent would be 24 bits.

An out-of-range value reported for an Array item indicates that no controls are asserted.

Absolute | Relative. Absolute means that the value is based on a fixed origin. Relative means that the data indicates the change from the last reading. A joystick normally reports absolute data (the joystick's current position), while a mouse reports relative data (how far the mouse has moved since the last report).

No Wrap | Wrap. Wrap indicates that the value rolls over to the minimum if the value continues to increment after reaching its maximum and that the value rolls over to the maximum if the value continues to decrement after reaching its minimum. An item specified as No Wrap that exceeds the spec-

ified limits may report a value outside the limits. This bit doesn't apply to Array data.

Linear | Non-linear. Linear indicates that the measured data and the reported value have a linear relationship. In other words, a graph of the reported data and the property being measured forms a straight line. In non-linear data, a graph of the reported data and the property being measured forms a curve. This bit doesn't apply to Array data.

Preferred State | No Preferred State. Preferred state indicates that the control will return to a particular state when the user isn't interacting with it. A momentary pushbutton has a preferred state (out) when no one is pressing the button. A toggle switch has no preferred state and remains in the last state selected by a user. This bit doesn't apply to Array data.

No Null Position | Null State. Null state indicates that the control supports a state where the control isn't sending meaningful data. A control indicates that it's in the null state by sending a value outside the range defined by its Logical Minimum and Logical Maximum. No Null Position indicates that any data sent by the control is meaningful data. A hat switch on a joystick is in a null position when it isn't being pressed. This bit doesn't apply to Array data.

Non-volatile | Volatile. The Volatile bit applies only to Output and Feature report data. Volatile means that the device can change the value on its own, without host interaction, as well as when the host sends a report requesting the device to change the value. For example, a control panel may have a control that users can set in two ways. A user may use a mouse to click a button on the screen to cause the host to send a report to the device, or a user may press a physical button on the device. Non-volatile means that the device changes the value only when the host requests a new value in a report.

When the host is sending a report and doesn't want to change a volatile item, the value to assign to the item depends on whether the data is defined as relative or absolute. If a volatile item is defined as relative, a report that assigns a value of 0 should result in no change. If a volatile item is defined as absolute, a report that assigns an out-of-range value should result in no change.

This bit doesn't apply to Array data.

Bit Field | Buffered Bytes. Bit Field means that each bit or a group of bits in a byte can represent a separate piece of data and the byte doesn't represent a single quantity. The application interprets the contents of the field. Buffered Bytes means that the data consists of one or more bytes. The report size for Buffered Byte items must be eight. This bit doesn't apply to Array data. Note that this bit is bit 8 in the item's data so using this bit requires two data bytes in the item.

Collection and End Collection Items

All of the report types can use Collection and End Collection items to group related items.

The three defined types of collections are application, physical, and logical. Vendors can also define collection types. Collections can be nested. Table 12-2 shows the values of the Collection and End Collection tags and the defined values for the different collection types.

An application collection contains items that have a common purpose or that together carry out a single function. For example, the boot descriptor for a keyboard groups the keypress and LED data in an application collection. All report items must be in an application collection.

A physical collection contains items that represent data at a single geometric point. A device that collects a variety of sensor readings from multiple locations might group the data for each location in a physical collection. The boot descriptor for a mouse groups the button and position indicators in a physical collection.

A logical collection forms a data structure consisting of items of different types that are linked by the collection. An example is the contents of a data buffer and a count of the number of bytes in the buffer.

Each collection begins with a Collection item and ends with an End Collection item. All Main items between the Collection and End Collection items are part of the collection. Each collection must have a Usage tag (described below).

Table 12-2: Data values for the Collection and End Collection Main Item Tags.

Main Item Type	Value	Description
Collection (A1h)	00h	Physical
	01h	Application
	02h	Logical
	03h-7Fh	Reserved
	80h-FFh	Vendor-defined
End Collection (C0h)	None	Closes a collection

A top-level collection is a collection that isn't nested within another collection. A HID interface can have more than one top-level collection, with each top-level collection representing a different HID. For example, a keyboard with an embedded pointing device can have a HID interface with two top-level collections, one for the pointing-device's reports and one for the keyboard's reports. Unlike HIDs in separate interfaces, these HIDs share interrupt endpoints.

If a report contains an unknown vendor-defined collection type, the host should ignore all Main items in the collection. If a known collection type has an unknown Usage, the host should ignore all items in the collection.

The Global Item Type

Global items identify reports and describe the data in them, including characteristics such as the data's function, maximum and minimum allowed values, and the size and number of report items. A Global item tag applies to every item that follows until the next Global tag. To save storage space in the device, the report descriptor doesn't have to repeat values that don't change from one item to the next. There are 12 defined Global items, shown in Table 12-3. The following sections describe the items in more detail.

Identifying the Report

Report ID. This value can identify a specific report. A HID can support multiple reports of the same type, with each report having its own report

Table 12-3: There are twelve defined Global items.

Global Item Type	Value (nn indicates the number of bytes that follow)	Description
Usage Page	000001nn	Specifies the data's usage or function.
Logical Minimum	000101nn	Smallest value that an item will report.
Logical Maximum	001001nn	Largest value that an item will report.
Physical Minimum	001101nn	The logical minimum expressed in physical units.
Physical Maximum	010001nn	The logical maximum expressed in physical units.
Unit exponent	010101nn	Base 10 exponent of units.
Unit	011001nn	Unit values.
Report Size	011101nn	Size of an item's fields in bits.
Report ID	100001nn	Prefix that identifies a report.
Report Count	100101nn	The number of data fields for an item.
Push	101001nn	Places a copy of the global item state table on the stack.
Pop	101101nn	Replaces the item state table with the last structure pushed onto the stack.
Reserved	110001nn to 111101nn	For future use.

ID, contents, and format. This way, a transfer doesn't have to include every piece of data every time. Often, however, the simplicity of having a single report is more important than the need to reduce the bandwidth used by longer reports.

In a report descriptor, a Report ID item applies to all items that follow until the next Report ID. If there is no Report ID item, the default ID of zero is assumed. A descriptor should not declare a Report ID of zero. The Report IDs are specific to each report type, so a HID can have one report of each type with the default ID. However, if at least one report type uses multiple report IDs, every report in the HID must have a declared ID. For example, if an interface supports Report ID 1 and Report ID 2 for Feature reports, any Input or Output reports must also have a Report ID greater than zero.

In a transfer that uses a Set_Report or Get_Report request, the host specifies a report ID in the Setup transaction, in the low byte of the wValue field. In

an interrupt transfer, if the interface supports more than one report ID, the report ID precedes the report data on the bus. If the interface supports only the default report ID of zero, the report ID isn't sent with the report when using interrupt transfers.

Under Windows, the report buffer provided to an API call must be large enough to hold the report plus one byte for the report ID. When a HID supports multiple report IDs for Input reports of different sizes, the Windows HID driver requires applications to use a buffer large enough to hold the longest report. When the HID supports multiple reports of the same type and different sizes and the HID is sending a report whose data is a multiple of the endpoint's maximum packet size, the HID indicates the end of the report data by sending a zero-length data packet

For Input reports when there are multiple Input Report IDs, the host's driver has no way to request a specific report from the HID. On receiving the IN token packet, the device returns whatever report is in its buffer. In other words, the device firmware decides which report to send. At the host, the HID driver stores the received report and its report ID in a buffer.

Describing the Data's Use

The Global items that describe the data and how it will be used are Usage Page, Logical and Physical Maximums and Minimums, Unit, and Unit Exponent. Each of these items helps the receiver of the report interpret the report's data. All but the Usage Page are involved with converting raw report data to values with units attached. The items make it possible for a report to contain data in a compact form, with the receiver of the data responsible for converting the data to meaningful values. However, the sender of the report data may instead choose to do some or all of the converting.

Usage Page. An item's Usage is a 32-bit value that identifies a function that a device performs. A Usage contains two values: the upper 16 bits are a Global Usage Page item and the lower 16 bits are a Local Usage item. The value in the Local Usage item is a Usage ID. The term *Usage* can refer to either the 32-bit value or the 16-bit Local value. To prevent confusion, some sources use the term Extended Usage to refer to the 32-bit value. In Microsoft's doc-

umentation, the USAGE type is a 16-bit value that can contain a Usage Page or a Usage ID.

Multiple items can share a Usage Page while having different Usage IDs. After a Usage Page appears in a report, all Usage IDs that follow use that Usage Page until a new Usage Page is declared.

The HID Usage Tables document lists the defined Usage Pages and their values and also names the section or other document that describes each page and its indexes. There are Usage Pages for many common device types, including generic desktop controls (mouse, keyboard, joystick), digitizer, bar-code scanner, camera control, and various game controls. In specialized devices that don't have a defined Usage Page, a vendor can define the Usage Page using values from FF00h to FFFFh.

Logical Minimum and Logical Maximum. The Logical Minimum and Logical Maximum define the limits for reported values. The limits are expressed in "logical units," which means that they use the same units as the values they describe. For example, if a device reports readings of up to 500 milliamperes in units of 2 milliamperes, the Logical Maximum is 250.

If the most significant bit of the highest byte is 1, the value is negative, expressed as a two's complement. (To find the negative value represented by a two's complement, complement each bit and add 1 to the result.) Using 1-byte values, 00h to 7Fh are the positive decimal values 0 through 127, and FFh to 80h are the negative decimal values -1 through -128.

The HID specification says that if both the Logical Minimum and Logical Maximum are considered positive, there's no need for a sign bit. But the report-descriptor test in the USB-IF Compliance Tool assumes that if the most-significant bit is 1, the value is negative. These values will fail the compliance test because the Logical Minimum (0) is greater than the Logical Maximum (-1):

```
0x15 0x00      // Logical Minimum
0x25 0xFF      // Logical Maximum WRONG!
```

If the desired result is a minimum of zero and a maximum of 255, the solution is to use a 2-byte value for the maximum:

```
0x15 0x00      // Logical Minimum
0x26 0x00FF    // Logical Maximum
```

Note that the Logical Maximum item tag is now 26h to indicate that the data that follows the tag is two bytes. Because the most-significant bit of the Logical Maximum is zero, the value is assumed to be positive and the compliance test accepts the values as valid.

Converting Units

The Physical Minimum, Physical Maximum, Unit Exponent, and Unit items define how to convert reported values into more meaningful units.

Physical Minimum and Physical Maximum. The Physical Minimum and Physical Maximum define the limits for a value when expressed in the units defined by the Units tag. In the earlier example of values of 0 through 250 in units of 2 milliamperes, the Physical Minimum is 0 and the Physical Maximum is 500. The receiving device uses the logical and physical limit values to obtain the value in the desired units. In the example, reporting the data in units of 2 milliamperes means that the value can transfer in a single byte, with the receiver of the data using the Physical Minimum and Maximum values to translate to milliamperes. The price is a loss in resolution, compared to reporting 1 bit per milliampere. If the report doesn't specify the values, they default to the same as the Logical Minimum and Logical Maximum.

Unit Exponent. The Unit Exponent specifies what power of 10 to apply to the value obtained after using the logical and physical limits to convert the value into the desired units. The exponent can range from -8 to +7. A value of 0 causes the value to be multiplied by 10^0, or 1, which is the same as applying no exponent. These are the codes:

Exponent	0	1	2	3	4	5	6	7	-8	-7	-6	-5	-4	-3	-2	-1
Code	00h	01h	02h	03h	04h	05h	06h	07h	08h	09h	0Ah	0Bh	0Ch	0Dh	0Eh	0Fh

For example, if the value obtained is 1234 and the Unit Exponent is 0Eh, the final value is 12.34.

Unit. The Unit tag specifies what units to apply to the report data after the value is converted using the Physical and Unit Exponent items. The HID specification defines codes for the basic units of length, mass, time, temperature, current, and luminous intensity. Most other units can be derived from these.

Specifying a Unit value can be more complicated than you might expect. Table 12-4 shows values you can work from. The value can be as long as four bytes, with each nibble having a defined function. Nibble 0 (the least significant nibble) specifies the measurement system, either English or SI (International System of Units) and whether the measurement is in linear or angular units. Each of the nibbles that follow represents a quality to be measured, with the value of the nibble representing the exponent to apply to the value. For example, a nibble with a value of 2 means that the corresponding value is in units squared. A nibble with a value of Dh, which represents -3, means that the units are expressed as $1/units^3$. These exponents are separate from the Unit Exponent value, which is a power of ten applied to the data, rather than an exponent applied to the units.

Converting Raw Data

To convert raw data to values with units attached, three things must occur. The firmware's report descriptor must contain the information needed for the conversion. The sender of the data must send data that matches the specification in the descriptor. And the receiver of the data must apply the conversions specified in the descriptor.

Below are examples of descriptors and raw and converted data. Remember that just because a tag exists in the HID specification doesn't mean you have to use it. If the application knows what format and units to use for the values it's going to send or receive, the firmware doesn't have to specify these items.

To specify time in seconds, up to a minute, the report descriptor might include this information:

Logical Minimum: 0
Logical Maximum: 60

Table 12-4: The units to apply to a reported value are a function of the measuring system and exponent values specified in the Unit item

Nibble Number	Quality Measured	Measuring System (Nibble 0 value)				
		None (0)	SI Linear (1)	SI Rotation (2)	English Linear (3)	English Rotation (4)
1	Length	None	Centimeters	Radians	Inches	Degrees
2	Mass	None	Grams		Slugs	
3	Time	None	Seconds			
4	Tempera-ture	None	Kelvin		Fahrenheit	
5	Current	None	Amperes			
6	Luminous Intensity	None	Candelas			
7	Reserved	None				

Physical Minimum: 0

Physical Maximum: 60

Unit: 1003h. Nibble 0 = 3 to select the English Linear measuring system (though in this case, any value from 1 to 4 would work). Nibble 3 = 1 to select time in seconds.

Unit Exponent: 0

With this information, the receiver knows that the value sent equals a number of seconds.

Now, what if instead you want to specify time in tenths of seconds, again up to a minute? You would need to increase the Logical and Physical Maximums and change the Unit Exponent:

Logical Minimum: 0

Logical Maximum: 600

Physical Minimum: 0

Physical Maximum: 600

Unit: 1003h. Nibble 0 = 3 to select the English Linear measuring system. Nibble 3 = 1 to select time in seconds.

Unit Exponent: 0Fh. This represents an exponent of -1, to indicate that the value is expressed in tenths of seconds rather than seconds.

Sending values as large as 600 will require 3 bytes, which the firmware specifies in the Report Size tag.

To send a temperature value using one byte to represent temperatures from -20 to 110 degrees Fahrenheit, the report descriptor might contain the following:

Logical Minimum: -128 (80h when expressed in hexadecimal as a two's complement)

Logical Maximum: 127 (7Fh)

Physical Minimum: -20 (ECh when expressed in hexadecimal as a two's complement)

Physical Maximum: 110 (6Eh)

Unit: 10003h. Nibble 0 is 3 to select the English Linear measuring system, though in this case, any value from 1 to 4 is OK. Nibble 4 is 3 to select degrees Fahrenheit.

Unit Exponent: 0

These values ensure the highest possible resolution for a single-byte report item, because the transmitted values can span the full range from 0 to 255.

In this case the logical and physical limits differ, so converting is required. This Visual-Basic code finds the resolution, or number of bits per unit:

```
Resolution = _
  (Logical_Maximum - Logical_Minimum) / _
  ((Physical_Maximum - Physical_Minimum) * _
  (10 ^ Unit_Exponent))
```

With the example values, the resolution is 1.96 bits per degree, or 0.51 degree per bit.

This Visual-Basic code converts a value to the specified units:

```
Value = _
  Value_In_Logical_Units * _
  ((Physical_Maximum - Physical_Minimum) * _
  (10 ^ Unit_Exponent )) / _
  (Logical_Maximum - Logical_Minimum)
```

If the value in logical units (the raw data) is 63, the converted value in the specified units is 32 degrees Fahrenheit.

As another example, specifying velocity in centimeters per second requires a Unit value that contains units of both centimeters and seconds. From Table 12-4, the Unit value to use is 1011h. Nibble 0 = 1 to select the SI measuring system, nibble 1 = 1 to select length in centimeters, and nibble 3 = 1 to select time in seconds.

To show how complicated it can get, the Unit value for volts is F0D121h, which indicates the SI Linear measuring system in units of $(cm^2)(gm)/(sec^{-3})(amp^{-1})$. However, remember that the Unit value only specifies the units. All the receiver has to do is identify the Units value and assign the units to received data; there's no need to do the calculations implied in the Units value.

Describing the Data's Size and Format

Two Global items describe the size and format of the report data.

Report Size specifies the size in bits of a field in an Input, Output, or Feature item. Each field contains one piece of data.

Report Count specifies how many fields an Input, Output, or Feature item contains.

For example, if a report has two 8-bit fields, Report Size is 8 and Report Count is 2. If a report has one 16-bit field, Report Size is 16 and Report Count is 1.

A single Input, Output, or Feature report can have multiple items, each with its own Report Size and Report Count.

Saving and Restoring Global Items

The final two Global items enable saving and restoring sets of Global items. These items allow flexibility in the report formats while using minimum storage space in the device.

Push places a copy of the Global-item state table on the CPU's stack. The Global-item state table contains the current settings for all previously defined Global items.

Pop is the complement to Push. It restores the saved states of the previously pushed Global item states.

The Local Item Type

Local items specify qualities of the controls and data items in a report. A Local item's value applies to all items that follow within a Main item until a new value is assigned. Local items don't carry over to the next Main item; each Main item begins fresh, with no Local items defined.

Local items relate to general usages, body-part designators, and strings. A Delimiter item enables grouping sets of Local items. Table 12-5 shows the values and meaning of each of the items.

Usage. The Local Usage item is the Usage ID that works together with the Global Usage Page to describe the function of a control, data, or collection.

As with the Usage Page item, the HID Usage Tables document lists many Usage IDs. For example, the Buttons Usage Page uses Local Usage IDs from 1 to FFFFh to identify which buttons in a set is pressed, with a value of 0 meaning no button pressed.

A report may assign one Usage to multiple items. If a report item is preceded by a single Usage, that Usage applies to all of the item's data. If a report item is preceded by more than one Usage and the number of controls or data items equals the number of Usages, each Usage applies to one control or data item, with the Usages and controls/data items pairing up in sequence. In the following example, the report contains two bytes. The first byte's Usage is X, and the second byte's Usage is Y.

```
Usage (X),
Usage (Y),
Report Count (2),
Report Size (8),
Input (Data, Variable, Absolute),
```

If a report item is preceded by more than one Usage and the number of controls or data items is greater than the number of Usages, each Usage pairs up with one control or data item in sequence, and the final Usage applies to all

Table 12-5: There are ten defined Local items.

Local Item Type	Value (nn indicates the number of bytes that follow)	Description
Usage	000010nn	An index that describes the use for an item or collection.
Usage Minimum	000110nn	The starting Usage associated with the elements in an array or bitmap.
Usage Maximum	001010nn	The ending Usage associated with the elements in an array or bitmap.
Designator Index	001110nn	A Designator value in a physical descriptor. Indicates what body part applies to a control.
Designator Minimum	010010nn	The starting Designator associated with the elements in an array or bitmap.
Designator Maximum	010110nn	The ending Designator associated with the elements in an array or bitmap.
String Index	011110nn	Associates a string with an item or control.
String Minimum	100010nn	The first string index when assigning a group of sequential strings to controls in an array or bitmap.
String Maximum	100110nn	The last string index when assigning a group of sequential strings to controls in an array or bitmap.
Delimiter	101010nn	The beginning (1) or end (0) of a set of Local items.
Reserved	101011nn to 111110nn	For future use.

of the remaining controls/data items. In the following example, the report is 16 bytes. Usage X applies to the first byte, Usage Y applies to the second byte, and a vendor-defined Usage applies to the third through 16th bytes.

```
Usage (X)
Usage (Y)
Usage (vendor defined)
Report Count (16),
Report Size (8),
Input (Data, Variable, Absolute)
```

Usage Minimum and Maximum. The Usage Minimum and Usage Maximum can assign a series of Usage IDs to the elements in an array or bitmap. The following example describes a report that contains the state (0 or 1) of each of three buttons. The Usage Minimum and Usage Maximum specify that the first button has a Usage ID of 1, the second button has a Usage ID of 2, and the third button has a Usage ID of 3:

```
Usage Page (Button Page)
Logical Minimum (0)
Logical Maximum (1)
Usage Minimum (1)
Usage Maximum (3)
Report Count (3)
Report Size (1)
Input (Data, Variable, Absolute)
```

Designator Index. For items with a physical descriptor, the Designator Index specifies a Designator value in a physical descriptor. The Designator specifies what body part the control uses.

Designator Minimum and Designator Maximum. When a report contains multiple Designator Indexes that apply to the elements in a bitmap or array, a Designator Minimum and Designator Maximum can assign a sequential Designator Index to each bit or array item.

String Index. An item or control can include a String Index to associate a string with the item or control. The strings are stored in the same format described in Chapter 4 for product, manufacturer, and serial-number strings.

String Minimum and Maximum. When a report contains multiple string indexes that apply to the elements in a bitmap or array, a String Minimum and String Maximum can assign a sequential String Index to each bit or array item.

Delimiter. A Delimiter defines the beginning (1) or end (0) of a local item. A delimited local item may contain alternate usages for a control. Different applications can thus define a device's controls in different ways. For example, a button may have a generic use (Button1) and a specific use (Send, Quit, etc.).

Physical Descriptors

A physical descriptor specifies the part or parts of the body intended to activate a control. For example, each finger might have its own assigned control. Similar physical descriptors are grouped into a physical descriptor set. A set consists of a header, followed by the physical descriptors.

A physical descriptor is a HID-specific descriptor. The host can retrieve a physical descriptor set by sending a Get_Descriptor request with 23h in the high byte of the wValue field and the number of the descriptor set in the low byte of the wValue field.

Physical descriptors are optional. For most devices, these descriptors either don't apply at all or the information they could provide has no practical use. The HID specification has more information on how to use physical descriptors, for those devices that need them.

Padding

To pad a descriptor so it contains a multiple of eight bits, a descriptor can include a Main item with no assigned Usage. The following excerpt from a keyboard's report descriptor specifies an Output report that transfers five bits of data and three bits of padding:

```
Usage Page (LEDs)
Usage Minimum (1)
Usage Maximum (5)
Output (Data, Variable, Absolute) (5 1-bit LEDs)

Report Count (1)
Report Size (3)
Output (Constant) (3 bits of padding)
```