

# INTERNATIONAL STANDARD

**ISO/IEC**  
**9636-5**

First edition  
1991-12-15

---

---

## **Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification —**

### **Part 5: Input and echoing**

*Technologies de l'information — Infographie — Interfaces pour  
l'infographie — Spécifications fonctionnelles —  
Partie 5: Entrée et résonance*



Reference number  
ISO/IEC 9636-5:1991(E)



**Documents**

Document Sharing Hub



**Documents**  
Document Sharing Hub

|        |                                                                |    |
|--------|----------------------------------------------------------------|----|
| 5.2.3  | ECHO CONTROLS .....                                            | 24 |
| 5.2.4  | PUT CURRENT <input class> MEASURE .....                        | 25 |
| 5.2.5  | ECHO DATA .....                                                | 26 |
| 5.2.6  | <input class> DEVICE DATA .....                                | 27 |
| 5.2.7  | ASSOCIATE TRIGGERS .....                                       | 30 |
| 5.2.8  | GET ADDITIONAL STROKE DATA .....                               | 30 |
| 5.2.9  | GET ADDITIONAL PICK DATA .....                                 | 31 |
| 5.2.10 | GET ADDITIONAL STRING DATA .....                               | 31 |
| 5.2.11 | GET ADDITIONAL RASTER DATA .....                               | 32 |
| 5.3    | Request and sample functions .....                             | 32 |
| 5.3.1  | REQUEST <input class> .....                                    | 32 |
| 5.3.2  | SAMPLING STATE .....                                           | 34 |
| 5.3.3  | SAMPLE <input class> .....                                     | 34 |
| 5.4    | Echo request input functions .....                             | 35 |
| 5.4.1  | INITIALIZE ECHO REQUEST .....                                  | 35 |
| 5.4.2  | ECHO REQUEST <input class> .....                               | 36 |
| 5.5    | Event input functions .....                                    | 38 |
| 5.5.1  | INITIALIZE EVENT QUEUE .....                                   | 38 |
| 5.5.2  | RELEASE EVENT QUEUE .....                                      | 38 |
| 5.5.3  | ENABLE EVENTS .....                                            | 39 |
| 5.5.4  | DISABLE EVENTS .....                                           | 39 |
| 5.5.5  | EVENT QUEUE BLOCK CONTROL .....                                | 40 |
| 5.5.6  | FLUSH EVENTS .....                                             | 40 |
| 5.5.7  | FLUSH DEVICE EVENTS .....                                      | 40 |
| 5.5.8  | AWAIT EVENT .....                                              | 41 |
| 5.5.9  | DEQUEUE <input class> EVENT .....                              | 42 |
| 5.5.10 | EVENT QUEUE TRANSFER .....                                     | 43 |
| 5.6    | Echo output functions .....                                    | 43 |
| 5.6.1  | INITIALIZE ECHO OUTPUT .....                                   | 43 |
| 5.6.2  | RELEASE ECHO OUTPUT .....                                      | 44 |
| 5.6.3  | ECHO OUTPUT CONTROLS .....                                     | 44 |
| 5.6.4  | PERFORM ACKNOWLEDGEMENT .....                                  | 45 |
| 5.6.5  | UPDATE <input class> ECHO OUTPUT .....                         | 45 |
| 5.6.6  | ECHO OUTPUT DATA .....                                         | 46 |
| 6      | Input and echoing inquiry functions .....                      | 48 |
| 6.1    | Introduction .....                                             | 48 |
| 6.1.1  | Data types employed .....                                      | 48 |
| 6.1.2  | Validity of returned information .....                         | 48 |
| 6.2    | Input description table .....                                  | 48 |
| 6.2.1  | INQUIRE INPUT CAPABILITY .....                                 | 48 |
| 6.2.2  | INQUIRE LIST OF AVAILABLE INPUT DEVICES .....                  | 48 |
| 6.3    | Class-independent logical input device description table ..... | 49 |
| 6.3.1  | INQUIRE COMMON INPUT DEVICE PROPERTIES .....                   | 49 |
| 6.3.2  | INQUIRE LIST OF SUPPORTED ECHO TYPES .....                     | 49 |
| 6.3.3  | INQUIRE LIST OF SUPPORTED PROMPT TYPES .....                   | 49 |
| 6.3.4  | INQUIRE LIST OF SUPPORTED ACKNOWLEDGEMENT TYPES .....          | 50 |
| 6.3.5  | INQUIRE LIST OF ASSOCIABLE TRIGGERS .....                      | 50 |
| 6.4    | Class-specific logical input device description table .....    | 50 |
| 6.4.1  | INQUIRE LOCATOR CAPABILITIES .....                             | 50 |
| 6.4.2  | INQUIRE STROKE CAPABILITIES .....                              | 51 |
| 6.4.3  | INQUIRE CHOICE CAPABILITIES .....                              | 51 |
| 6.4.4  | INQUIRE PICK CAPABILITIES .....                                | 51 |
| 6.4.5  | INQUIRE STRING CAPABILITIES .....                              | 51 |
| 6.4.6  | INQUIRE LIST OF AVAILABLE INPUT CHARACTER SETS .....           | 52 |
| 6.4.7  | INQUIRE RASTER INPUT CAPABILITIES .....                        | 52 |
| 6.4.8  | INQUIRE LIST OF PERMITTED RASTER SPOT CENTRE SEPARATIONS ..... | 52 |
| 6.4.9  | INQUIRE GENERAL CAPABILITIES .....                             | 52 |
| 6.4.10 | INQUIRE LIST OF SUPPORTED GENERAL MEASURE FORMATS .....        | 53 |
| 6.5    | Class-independent logical input device state list .....        | 53 |
| 6.5.1  | INQUIRE COMMON LOGICAL INPUT DEVICE STATE .....                | 53 |
| 6.5.2  | INQUIRE LIST OF ASSOCIATED TRIGGERS .....                      | 53 |

|     |        |                                                            |    |
|-----|--------|------------------------------------------------------------|----|
|     | 6.5.3  | INQUIRE ECHO DATA RECORD .....                             | 54 |
|     | 6.5.4  | INQUIRE INPUT DEVICE DATA RECORD .....                     | 54 |
| 6.6 |        | Class-specific logical input device state list .....       | 54 |
|     | 6.6.1  | INQUIRE LOCATOR STATE .....                                | 54 |
|     | 6.6.2  | INQUIRE STROKE STATE .....                                 | 55 |
|     | 6.6.3  | INQUIRE VALUATOR STATE .....                               | 55 |
|     | 6.6.4  | INQUIRE CHOICE STATE .....                                 | 56 |
|     | 6.6.5  | INQUIRE PICK STATE .....                                   | 56 |
|     | 6.6.6  | INQUIRE STRING STATE .....                                 | 56 |
|     | 6.6.7  | INQUIRE RASTER INPUT STATE .....                           | 57 |
|     | 6.6.8  | INQUIRE GENERAL STATE .....                                | 57 |
| 6.7 |        | Event input state list .....                               | 57 |
|     | 6.7.1  | INQUIRE EVENT INPUT STATE .....                            | 57 |
| 6.8 |        | Echo output description table .....                        | 58 |
|     | 6.8.1  | INQUIRE ECHO OUTPUT CAPABILITIES .....                     | 58 |
|     | 6.8.2  | INQUIRE LIST OF ECHO OUTPUT ECHO TYPES .....               | 58 |
|     | 6.8.3  | INQUIRE LIST OF ECHO OUTPUT PROMPT TYPES .....             | 58 |
|     | 6.8.4  | INQUIRE LIST OF ECHO OUTPUT ACKNOWLEDGEMENT TYPES .....    | 59 |
|     | 6.8.5  | INQUIRE LIST OF SUPPORTED GENERAL FORMAT IDENTIFIERS ..... | 59 |
| 6.9 |        | Echo entity state list .....                               | 59 |
|     | 6.9.1  | INQUIRE LIST OF CURRENTLY EXISTING ECHO ENTITIES .....     | 59 |
|     | 6.10   | Individual echo entity state list .....                    | 60 |
|     | 6.10.1 | INQUIRE ECHO ENTITY STATE .....                            | 60 |
|     | 6.10.2 | INQUIRE ECHO OUTPUT DATA RECORD .....                      | 60 |
| 7   |        | CGI description tables and state lists .....               | 61 |
|     | 7.1    | Description tables .....                                   | 61 |
|     | 7.1.1  | Input capability .....                                     | 61 |
|     | 7.1.2  | Class-independent logical input capability .....           | 61 |
|     | 7.1.3  | Echo output capability. ....                               | 63 |
|     | 7.2    | State lists .....                                          | 64 |
|     | 7.2.1  | Input state .....                                          | 64 |
|     | 7.2.2  | Class-independent logical input device state .....         | 64 |
|     | 7.2.3  | Class-specific logical input device state .....            | 65 |
|     | 7.2.4  | Events .....                                               | 67 |
|     | 7.2.5  | Echo entity state .....                                    | 68 |
| A   |        | Formal grammar of the functional specification .....       | 69 |
| B   |        | Input errors .....                                         | 92 |
| C   |        | Guidelines for CGI implementors .....                      | 94 |

## Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 9636-5 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*.

ISO/IEC 9636 consists of the following parts, under the general title *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification*:

- *Part 1: Overview, profiles, and conformance*
- *Part 2: Control*
- *Part 3: Output*
- *Part 4: Segments*
- *Part 5: Input and echoing*
- *Part 6: Raster*

Annexes A and B form an integral part of this part of ISO/IEC 9636. Annex C is for information only.

## Introduction

This part of ISO/IEC 9636 describes the interface between a device-independent part of a graphics system and INPUT or OUTIN Virtual Devices from which graphical and non-graphical inputs can be obtained.

CGI input functions control and perform different methods of input from a Virtual Device, allowing the return of an input value according to user requirement.

CGI input functions return a variety of data types which correspond to output data commonly used in graphics systems.

# Information technology – Computer graphics – Interfacing techniques for dialogues with graphical devices (CGI) – Functional specification –

## Part 5: Input and echoing

### 1 Scope

This part of ISO/IEC 9636 defines those functions of the Computer Graphics Interface concerned with obtaining graphical and non-graphical input from a Virtual Device of device class INPUT or OUTIN. This part of ISO/IEC 9636 also defines functions to support echoing of input operations on separate Virtual Devices.

This part of ISO/IEC 9636 is part 5 of ISO/IEC 9636, and should be read in conjunction with ISO/IEC 9636-1, ISO/IEC 9636-2, and ISO/IEC 9636-4. The relationship of this part of ISO/IEC 9636 to the other parts of of ISO/IEC 9636 is described in ISO/IEC 9636-1 and in clause 4.

## 2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9636. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 9636 are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 9636-1 : 1991 *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification — Part 1: Overview, profiles, and conformance.*

ISO/IEC 9636-2 : 1991 *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification — Part 2: Control.*

ISO/IEC 9636-3 : 1991 *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification — Part 3: Output.*

ISO/IEC 9636-4 : 1991 *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification — Part 4: Segments.*

ISO/IEC 9636-6 : 1991 *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Functional specification — Part 6: Raster.*

ISO/IEC 9637-1 : -<sup>1)</sup> *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Data stream binding — Part 1: Character encoding.*

ISO/IEC 9637-2 : -<sup>1)</sup> *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Data stream binding — Part 2: Binary encoding.*

ISO/IEC TR 9973 : 1988 *Information processing — Procedures for registration of graphical items.*

---

<sup>1)</sup> To be published.

## 3 Concepts

### 3.1 Introduction

This part of ISO/IEC 9636 defines those functions of the Computer Graphics Interface concerned with input and echoing. This functionality is divided into the following areas

- *Input control functions*, which provide control over initialization and deallocation of Logical Input Devices (LIDs), together with the means to tailor their characteristics.
- *Request and sample functions*, which permit LIDs to be used with Request and Sample input methods.
- *Echo request input functions*, which permit LIDs to be used with the Echo Request input method, a special type of Request method in which changes to the LID's measure value can be tracked by the client.
- *Event input functions*, which permit LIDs to be used with the Event input method; this allows the client to control a number of active LIDs simultaneously while concurrently performing graphical output.
- *Echo output functions*, which allow values to be echoed on a given CGI Virtual Device when the source of such values was not the given device.
- *Input and echoing inquiry functions*, which provide access to the description tables and state lists concerned with input and echoing.

### 3.2 Basic input model

The functions specified by this part of ISO/IEC 9636 are defined in terms of a model of the operation of graphical input. This sub-clause defines the relationship between the top level concepts involved, and the following sub-clauses detail each of the elements of the input model.

An INPUT or OUTIN CGI Virtual Device provides input capabilities through *Logical Input Devices (LIDs)*.

Input may be acquired by means of different *input methods*, and the client of the CGI may also exert control of the operation of *prompt*, *echo*, and *acknowledgement* capabilities of the Virtual Device.

### 3.3 Logical input devices

A logical input device consists of a *measure*, *state information* and a set of associated *triggers*. (A LID's set of associated triggers may be empty.)

A LID is identified by the *input class* of its measure (see table 1) and by its *LID index*. There is a separate set of LID indices for each class of measure.

Each LID has a description table and a state list; each is divided into a class-independent and a class-specific portion. The Class-Independent Logical Input Device State List holds entries for the input device state, the sample input state, and the controls for prompt, echo and acknowledgement. The Class-Specific Logical Input Device State List defines the control that is applied to the LID's measure. Both the class-specific and the class-independent state lists form the Logical Input Device State List, which exists for every LID. It is not required that the list of available logical input device indices for a specific input class be dense.

Table 1 – Input Classes

| Input classes |                                                                                                 |                       |
|---------------|-------------------------------------------------------------------------------------------------|-----------------------|
| Input Class   | Data returned                                                                                   | Example               |
| LOCATOR       | a single VDC point                                                                              | digitizer, mouse      |
| STROKE        | a sequence of VDC points                                                                        | digitizer             |
| VALUATOR      | a number from a continuous range of alternatives                                                | potentiometer         |
| CHOICE        | an integer from a bounded range of alternatives                                                 | button box            |
| PICK          | a pick status, and the list of pick values, containing a pick identifier and segment identifier | light pen             |
| STRING        | a character string                                                                              | alphanumeric keyboard |
| RASTER        | an input colour value array                                                                     | scanner               |
| GENERAL       | a data record                                                                                   | voice input           |

### 3.4 Measures

A measure is the modelled counterpart of the real-world entity that generates values that feed into the CGI input mechanisms. For the purposes of the input model, a measure has three components, an *input tool*, a *current value* and a *measure validity*. Both the current value and the measure validity are interrogated by the LID as part of any input action; the measure validity status determines completeness and reliability of the measure. If the measure validity is VALID, the measure value is complete and reliable and may be returned. If the measure validity is INVALID, the current value of the measure may not be returned and no meaning should be attached to the returned parameter value.

In normal operation, the input tool modifies the current value and the measure validity under the influence of an outside agent (for example, an operator).

The CGI client may also set the current value and the measure validity, using the function PUT CURRENT <input class> MEASURE. Some physical implementations of measures may render implementation of this function impractical; this situation is indicated by the Put Current Measure Effective entry in the Class-Independent Logical Input Device Description Table.

#### 3.4.1 Measures of input class LOCATOR

The value of a LOCATOR measure is a point in VDC space. A separate VDC coordinate system, independent of that used for graphical output, may be specified for each LOCATOR LID. (See 3.5.)

The criterion for the validity of a LOCATOR measure is that the point measured should be within the Input Extent.

Measures of input class LOCATOR may be implemented, for example, by a mouse, trackball, tablet puck, joystick, thumb-wheels, and by digitizing tables.

#### 3.4.2 Measures of input class STROKE

The value of a STROKE measure is a list of points in VDC space. A separate VDC coordinate system, independent of that used for graphical output, may be specified for each STROKE LID. (See 3.5.)

There is no criterion for a STROKE measure to be invalid; points outside the Input Extent are rejected.

An implementation of a STROKE measure will typically involve several external actions before a measure value becomes available. Successive use of SAMPLE input from a STROKE measure will characteristically produce values that (at least in part) repeat one another. The “complete” measure value is that returned to a REQUEST or ECHO REQUEST input function, or that entered in the event queue when a trigger fires. After the complete measure value has been made available, use of SAMPLE will return an “empty” measure value until further external action has occurred.

Measures of input class STROKE may be implemented by the same sorts of devices as for class LOCATOR, with a (possibly) separate method of signalling “input complete”.

### 3.4.3 Measures of input class VALUATOR

The value of a VALUATOR measure is a real number in a range defined by maximum and minimum values specified by entries in the Class-Specific Logical Input Device State List.

The criterion for the validity of a VALUATOR measure is that its value should fall in the specified range.

Measures of input class VALUATOR may be implemented, for example, by a single thumb-wheel or by a potentiometer knob.

### 3.4.4 Measures of input class CHOICE

The value of a CHOICE measure is an integer in a range whose lower bound is 1 and whose upper bound is defined in the Class-Specific Logical Input Device Description Table.

The criterion for the validity of a CHOICE measure is that its value should fall in the specified range.

Measures of input class CHOICE may be implemented, for example, by a button box, by a tablet, or screen menu.

### 3.4.5 Measures of input class PICK

The value of a PICK measure is a list of pick values, (see the definition of data type PV in ISO/IEC 9636-1, 5.2.10). The implementation of input class PICK requires an OUTIN Virtual Device to support the capabilities defined in ISO/IEC 9636-4. A separate VDC coordinate system, independent of that used for graphical output, may be specified for each PICK LID. (See 3.5.) This VDC coordinate system is used to define the pick aperture relative to the pick location.

The criteria that define which segments are picked are defined in ISO/IEC 9636-4, 3.8. PICK measures may exist only on Virtual Devices of device class OUTIN, since they derive values from an interaction between an input tool and segment storage of the Virtual Device. The PICK measure is INVALID if the number of pick values exceeds the maximum specified in the Class-Specific Logical Input Device State List.

Measures of input class PICK may be implemented by the same sort of devices as for class LOCATOR with some means to link it to segment storage.

### 3.4.6 Measures of input class STRING

The value of a STRING measure is a string of characters which is subject to the Input Character Set Index, Alternate Input Character Set Index, and the Input Character Coding Announcer entries in the Class-Specific Logical Input Device State List.

The STRING measure is INVALID if the number of characters exceeds the maximum specified in the Class-Specific Logical Input Device State List.

An implementation of a STRING measure will typically involve several external actions before a measure value becomes available. Successive use of SAMPLE input from a STRING measure will characteristically produce values that (at least in part) repeat one another. The “complete” measure value is that returned to a REQUEST or ECHO REQUEST input function, or that entered in the event queue when a trigger fires. After the complete measure value has been made available, use of SAMPLE will return an “empty” measure value until further external action has occurred.

A measure of input class STRING may be implemented, for example, by means of a computer keyboard.

### 3.4.7 Measures of input class RASTER

The value of a RASTER measure is an array of input colour values (typically colour or grey-scale intensities of an external image). The representation of each colour value is determined by the entries Colour and Bits Per Colour in the Class-Specific Logical Input Device State List. The Source Window, specified in pixels, in the Client-Specific Logical Input Device State List, determines the region from which the returned input colour values are obtained. Invoking the function RASTER DEVICE DATA can be used to change the setting of this Source Window. The pixel offsets used to specify the source window may result in the return of an invalid RASTER measure.

The measure of input class RASTER may be implemented, for example, by a video camera, a scanner, facsimile machine, or remote-sensing equipment. A conforming measure of input class RASTER shall not be implemented using the contents of a CGI bitmap. (GET PIXEL ARRAY, defined in ISO/IEC 9636-6, can be used for this purpose.)

### 3.4.8 Measures of input class GENERAL

The value of a GENERAL measure is a data record, whose format is determined by a measure format identifier. The GENERAL input class provides a mechanism for addressing, within the CGI model of input, logical input devices which do not match one of the other standardized input classes. The GENERAL measure is INVALID if the data record size exceeds the maximum specified in the Class-Specific Logical Input Device State List.

Non-negative measure format identifiers for input class GENERAL are reserved for those registered with the ISO Registration Authority. Negative measure format identifiers may be used for implementation-dependent formats.

## 3.5 Coordinate systems for VDC measures

LOCATOR and STROKE devices deliver VDC measure values. A GENERAL input device may also deliver a VDC measure value. Although PICK devices do not deliver VDC measure values, pick input uses a pick aperture specified in VDC that is used in determining the list of pick values returned.

The characteristics of the physical device that supports such logical input devices appear in the Class-Specific Logical Input Device Description Table. These physical characteristics include:

- Input device address space. This is the native coordinate range of the input device, and is expressed as a pair of ISPs, being the extreme lower left and upper right corners of the address range.
- Input device physical size. This is expressed as a pair of real numbers which gives the dimensions of the device in millimetres. For certain types of device (e.g. mouse, trackball) this may refer to a nominal rectangle rather than an actual size.
- Input device resolution. This is expressed as the number of resolvable steps over the complete address range of each axis. This information is of an advisory nature, since the resolution may vary across a device's range and through its operational lifetime.

For device of class LOCATOR, STROKE, and PICK, the Class-Specific Logical Input Device Description Table specifies a Physical Input Surface Size entry, in millimetres, and an Input Surface Size Interpretation entry with possible values of NOMINAL, ACTUAL, and UNLIMITED. The value ACTUAL is used for input devices for which an exact physical size is meaningful (e.g. tablets). The value NOMINAL is used for input devices for which an effective bounding region of a conceptual approximate size is more appropriate (e.g. a mouse or trackball). The value UNLIMITED indicates that no bound is meaningful and the Physical Input Surface Size entry only relates to the interpretation of the Number of Distinguishable Steps in XY Directions entry in the Class Specific Logical Input Device Description Table.

Logical Input Devices of class LOCATOR, STROKE, and PICK all start from a point in ISC space which determines a point in VDC space. The ISC-to-VDC Mapping for such a LID maps points on its input surface to VDC space. In order to echo such a point for an operator, the resulting VDC point is transformed by the echo mapping associated with the LID.

The ISC-to-VDC Mapping is specified in terms of an Input Viewport in ISC space and an Input Extent for the LID. The Input Viewport is a parallelogram on the input surface. It is nominally a rectangle aligned with the ISC axes; however, provision is made for the possibility that this rectangle is rotated or skewed. The Input Viewport rectangle is parameterized by three ISPs, conceptually corresponding to bottom left, bottom right and top right. The resulting ISC-to-VDC Mapping is a transformation which maps the first point of the Input Viewport onto the first point of the Input Extent, the second point of the Input Viewport onto a VDC point with the same x-coordinate as the second point of the Input Extent, and the same y-coordinate as the first point of the Input Extent. The third point of the Input Viewport is mapped onto the second point of the Input Extent.

For devices of class OUTIN, the echo mapping is determined by the Input Extent for the LID and an Echo Viewport for the LID on the display surface. The Echo Viewport is specified as two VPs. The echo mapping is linear in each of x and y and maps the first point of the Input Extent onto the first point of the Echo Viewport, and similarly for the second points, see figure 1.

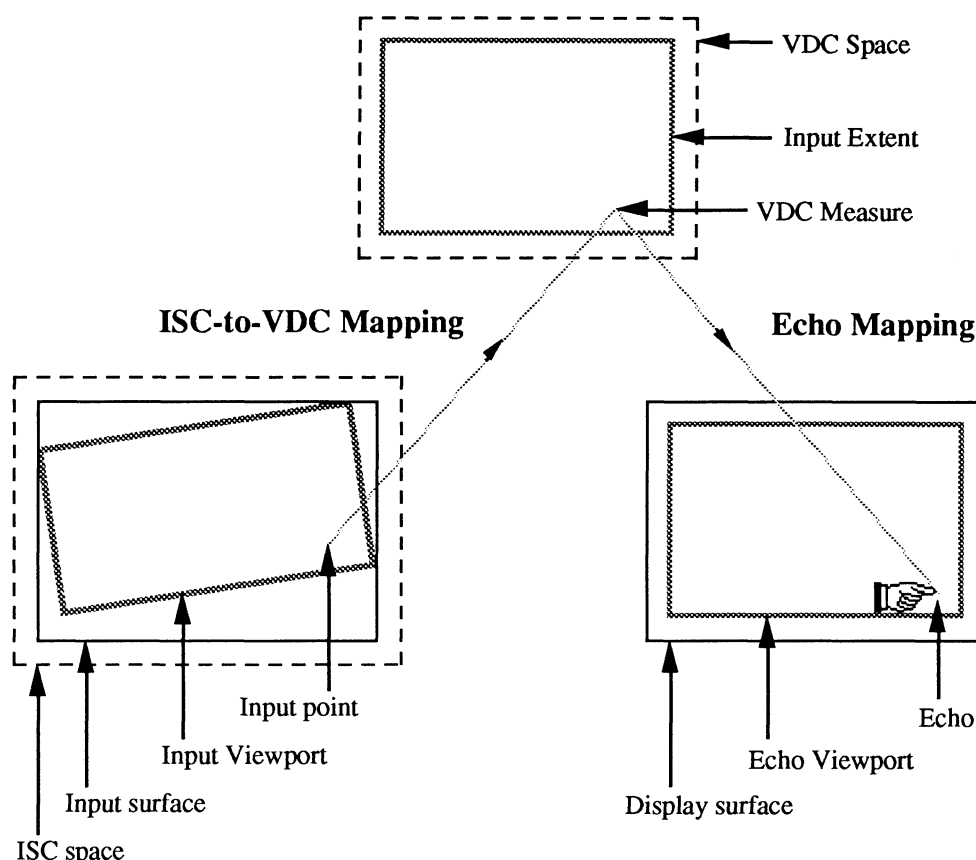


Figure 1 – Input Surface Coordinate mapping

When a LID with a VDC measure is initialized on a device of class OUTIN, the state list entries controlling its ISC-to-VDC Mapping and its echo mapping are initialized in such a way that the Input Viewport is mapped isotropically onto the entire display surface. This Input Viewport is initialized to the largest rectangle within the input surface, aligned at the lower left with respect to the input surface, having the same aspect ratio as the display surface, where both aspect ratios are measured in length units as perceived by an operator. The echo mapping (determined by the Input Extent and Echo Viewport state list entries for the LID) is initialized in such a way that the transformations for the echo mapping and the VDC-to-Device Mapping (for the display surface) are effectively the same, although they need not be parameterized in the same way. Specifically, the Echo Viewport is initialized to the entire display surface and the Input Extent is initialized to the image of this Echo Viewport under the inverse of the VDC-to-Device Mapping for the display surface.

The entries for Input Extent, Input Viewport and Echo Viewport are maintained in the Class-Specific Logical Input Device State List. Using the function <input class> DEVICE DATA, the client may redefine the ISC-to-VDC Mapping in a relatively unconstrained manner, as the Input Viewport is not required to be entirely inside the input surface. For example, the client might wish to map the entire input surface onto an Echo Viewport that is smaller than the display surface and accessed by (input) VDC points which bear no direct relation to VDCs being used for graphic output. Conversely, the client might map only a subset of the input surface onto the entire display surface.

For devices of class INPUT, when the LID is initialized the Input Extent is set to an implementation-dependent rectangle and the Input Viewport is set to the full range of the input device.

The Input Extent entry is interpreted in accordance with the VDC type and precision in effect at the time of its definition. Hence, changing the VDC type sets the Input Extent to the default for the chosen type.

The settings of the Device Viewport Specification Mode and Device Viewport Mapping have no relevance to the Input Viewport, which is always specified in native input coordinates (ISPs). Its mapping to the Input Extent is not necessarily isotropic.

For devices of class OUTIN, the position of a geometric echo on the display surface is determined by the current value of the measure. The appearance of an echo should be altered as its image in VDC passes outside of the Input Extent. This part of ISO/IEC 9636 allows latitude from this preferred behaviour and whether the preferred behaviour is supported is indicated by the Echo Change Support entry in the Class-Independent Logical Input Device Description Table. If the LID is sampled in such circumstances, the measure validity status is INVALID.

For devices of class OUTIN, the Echo Area rectangle, which is used to position certain types of non-geometric echo on the display surface, has no relevance to geometric echoing which automatically tracks the current value of a measure over the display surface.

## 3.6 Triggers

A trigger is typically (though not exclusively) a physical control manipulated by an operator. When a defined set of conditions is satisfied, the trigger is said to *fire*. The trigger firing defines the moment in time at which an input action takes place; if a LID with which the trigger is associated is enabled for request, echo request or event input, the current value of the LID's measure is stored, and thereby made available to the client. If the LID's Acknowledgement Control is ENABLED, then acknowledgement output is also performed when the trigger fires for request or echo request input or when the event report is enqueued for event input.

### 3.6.1 Trigger association

Triggers may exist independently of any particular LID. For trigger firing to have effect, it is necessary that the trigger be associated with at least one LID.

The CGI offers the client a mechanism for creating and deleting associations between triggers and LIDs. Each LID's Class-Independent Logical Input Device Description Table holds a list of trigger indices indicating those triggers that may be associated with the LID; a (possible empty) subset of this list is designated as triggers that are necessarily associated with the LID, and may not be dissociated. A single trigger may be associated with more than one LID; in this case, when the trigger fires each LID with which it is associated may *simultaneously* deliver input events to the client.

### 3.6.2 Timeouts

Some input functions offer a timeout capability. This occurs in situations where the function may not be able to return until the operator takes some action. In case the operator has left or is not paying attention, this provides a mechanism whereby control will eventually be returned to the client in any case.

The duration of a timeout interval is specified in seconds by a real number. A negative value of timeout indicates *wait forever*. That is, there is no possibility of the function returning due to an elapsed timeout interval.

Even if a Virtual Device cannot present a true timeout capability, this wait forever capability is always available. In these cases, a positive value indicates wait forever for the functions REQUEST and INITIALIZE ECHO REQUEST, and indicates a zero timeout for the function AWAIT EVENT. The Timeout Capability entry in the Input Device Description Table specifies the implemented timeout behaviour. The preferred behaviour is FULL timeout capability.

### 3.6.3 The break action

The operator may abort input by executing a *break action*. The break action cause the LID to abort any input, and to return a pseudo-event to the client, which has no valid measure associated with it. The precise effect of the break action on a LID depends on its input class.

There are two possibilities for the method of triggering a break action. A break action may be initiated by a single method that is common to the entire Virtual Device and applies over a multiplicity of LIDs (in which case, no LID class is associated with it). On the other hand, one may be initiated by a method that is specifically associated with a particular LID (e.g. one of the buttons on a mouse).

If the operator initiates a break action when a LID is in the state REQUEST PENDING or ECHO REQUEST PENDING, then the effect of that break is limited to the (single) LID in that state and the REQUEST or ECHO REQUEST soliciting function returns immediately. At other times, there may be multiple LIDs which are all active for sampling or event input, in which

case the effect of the break (discarding all partially constructed input) is applicable to all active LIDs of class STRING or STROKE.

## 3.7 Input methods and state model

### 3.7.1 Logical input device model

There are four methods of performing input with any logical input device of the input classes described above. These four methods are available through groups of functions for request, sample, event, and echo request input.

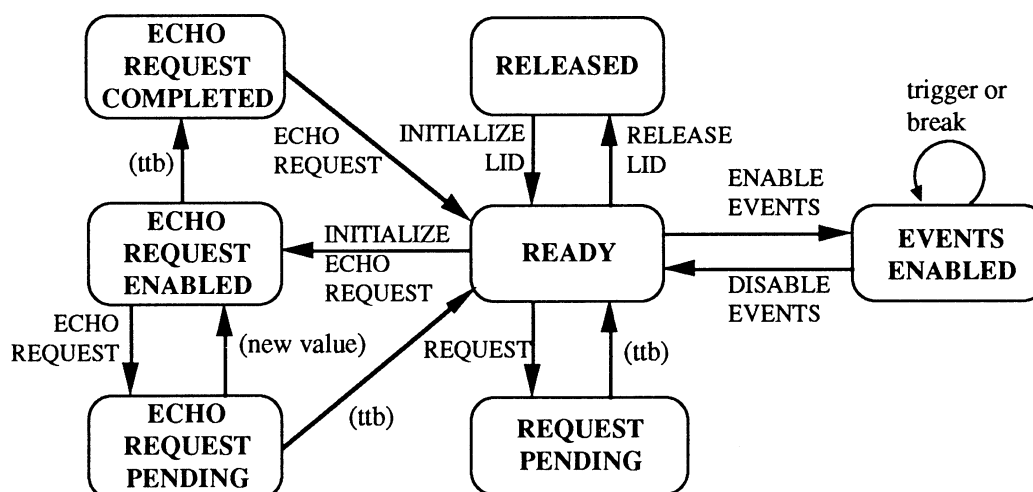
The interface to CGI input functions is a synchronous interface, although asynchronous actions may occur inside the Virtual Device.

The operating state of each logical input device is specified in the appropriate Class-Independent Logical Input Device State List by two input device state variables, the Input Device State and the Sampling State. In addition, the Event Queue State variable in the Event Queue State List defines further input states which are applicable when any input device is using the event input method.

The states in which it is not permissible to invoke the various input functions are defined in tables 13 and 14 and in terms of the logical input device state diagram, figure 2. This state diagram depicts transitions in the value of the Input Device State variable. The transitions take place either as a result of the client invoking a function, as a result of operator action, or as a result of certain other standardized conditions within the LID.

The Input Device State and the Sampling State (see 3.7.3) are defined separately for every instance of a logical input device. Thus, state restrictions apply on an individual basis for each logical input device.

The initial Input Device State of a LID is RELEASED and its measure is unavailable. A LID Input Device State will be set to state READY by the function INITIALIZE LOGICAL INPUT DEVICE. The function RELEASE LOGICAL INPUT DEVICE may be invoked in any state of the LID, the state after invoking RELEASE LOGICAL INPUT DEVICE is always RELEASED. The function INITIALIZE LOGICAL INPUT DEVICE may be invoked in any state of the LID, the state after invoking INITIALIZE LOGICAL INPUT DEVICE is always READY. Note, those transitions are not shown in the state diagrams to avoid unnecessary complexity in the figures.



ttb = trigger, timeout, or break

Figure 2 – Logical Input Device state diagram

There are state transitions from READY to other states, in which the LID is said to be *active* and which differ for the different input methods. In general, the LID is active if its Sampling State is ENABLED or if its Input Device State is anything other than READY or RELEASED. The transition to *active* is relevant for prompting and echo, (see 3.8). The state list of a LID may not be modified by the client except for its state variables when a LID is active. The states in which the LID is active are described in 3.7.2, 3.7.3, 3.7.4, and 3.7.5.

### 3.7.2 Request input

The invocation of REQUEST <input class> changes the LID's Input Device State from READY to REQUEST PENDING and, if the LID was not already active (due to sampling being enabled), the LID becomes active until the return to state READY. There are three reasons for a state transition back to state READY:

- A *trigger fires*: If the Acknowledgement Control is ENABLED, an acknowledgement is performed, the trigger index and the measure are returned and the request status is TRIGGER FIRED.
- If no trigger fires and no break occurs, the function returns after a time interval, which is defined in the timeout parameter, and the request status is TIMEOUT.
- The *break action* is performed: The returned measure is undefined and the request status is BREAK.

### 3.7.3 Sample input

The Sampling State variable of a logical input device, controls the use of the LID for the sample input method. The possible values for Sampling State are ENABLED and DISABLED (see figure 3). When the LID is initialized the Sampling State is set to DISABLED. The function SAMPLING STATE may be used to change the Sampling State to ENABLED. The LID is *active* at all times when its Sampling State is ENABLED. The Sampling State ENABLED may exist in parallel to other LID states which also activate the LID (i.e. when the LID is enabled for echo request or events, the Sampling State may also be ENABLED).

The invocation of SAMPLE <input class> returns the measure immediately. For input classes STRING and STROKE, the measure is returned as far as it is constructed at the time of invocation. If the device is also enabled for another input method, the buffers are cleared when an associated trigger fires or a break is performed. SAMPLE <input class> may also be called when no trigger is associated with the LID.

The function SAMPLING STATE may also be used to reset the Sampling State of a logical input device to DISABLED.

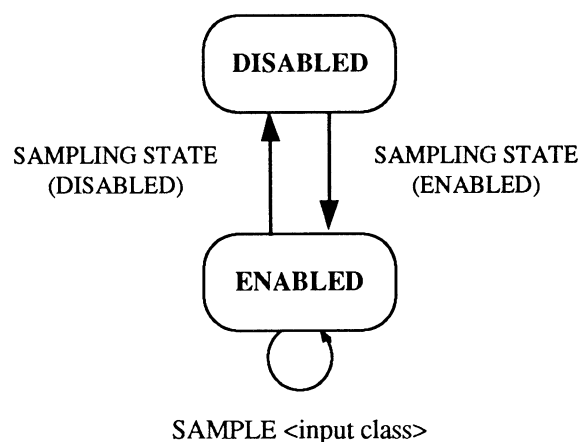


Figure 3 – Sampling State diagram

### 3.7.4 Remote echoing

The CGI provides functions to permit the echoing of the current measure value of a logical input device on one CGI Virtual Device during an echo request input, using the display surface of a different CGI Virtual Device (see figure 4). This other output device may also be used for prompting and acknowledgement output. This is referred to as *echo output*. For example, remote echoing might be used to echo the input from a digitizer on a physically separate graphics terminal display surface, since it may be useful or necessary for the digitizer and terminal to be treated as distinct Virtual Devices. (Note that echo

## Input methods and state model

## Concepts

output capability may be used whether or not the source of the data being echoed is a CGI implementation.) Prompt and acknowledgement can be performed by the INPUT device, the OUTPUT device, or both, dependent on the settings of the echo controls.

Remote Echoing enables a client of the CGI to manage echoing on the remote device in a manner which emulates the automatic echoing performed below the CGI for an OUTIN CGI Virtual Device in which echo output is directed to the same display surface as is being used for output.

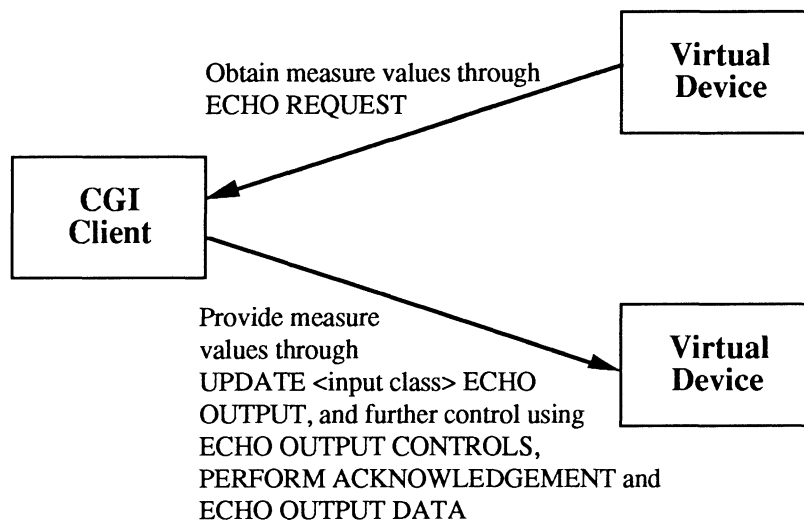


Figure 4 – Flow of data during remote echoing

A client performing echo output of an echo request input can initiate a timeout interval for a logical input device, then alternate between obtaining the current value of the measure and echoing this value on the output Virtual Device, until the timeout interval elapses. This differs from the basic request input method, since control is returned to the CGI client before the timeout interval has elapsed.

The following CGI input functions are provided in the Echo Request Input function group:

- INITIALIZE ECHO REQUEST
- ECHO REQUEST <input class>

The effect of INITIALIZE ECHO REQUEST is to initiate the timeout interval and arm triggers for the logical input device.

Unlike REQUEST, the function ECHO REQUEST has the property that it will return a new value to the client at such time as the measure value *changes*. No trigger, break, or timeout is required, although such events will also cause the ECHO REQUEST function to return. In the presence of Remote Echoing, the state diagram for a logical input device has three states associated with the ECHO REQUEST function:

- ECHO REQUEST ENABLED:** The logical input devices goes into this state after an INITIALIZE ECHO REQUEST.
- ECHO REQUEST COMPLETED:** The logical input device enters this state from ECHO REQUEST ENABLED on occurrence of a timeout, break, or the firing of a trigger.
- ECHO REQUEST PENDING:** The logical input device enters this state from ECHO REQUEST ENABLED in response to an ECHO REQUEST function. It is somewhat analogous to REQUEST PENDING. In particular, like REQUEST, the ECHO REQUEST function does not return until the state is exited, and the firing of a trigger will return the logical input device to the state READY. However, for the ECHO REQUEST PENDING state, a *change* in measure value returns the logical input device to the state ECHO REQUEST ENABLED.

## Concepts

## Input methods and state model

The following CGI output functions are provided in the Echo Output function group to control the echoing of output on a separate output Virtual Device:

- INITIALIZE ECHO OUTPUT
- RELEASE ECHO OUTPUT
- ECHO OUTPUT CONTROLS
- PERFORM ACKNOWLEDGEMENT
- UPDATE <input class> ECHO OUTPUT
- ECHO OUTPUT DATA

The process of performing echo output for a given stream of input data is controlled by means of a conceptual object which is referred to as an *echo output entity*. Conceptually, an echo output entity is created by INITIALIZE ECHO OUTPUT and deleted by RELEASE ECHO OUTPUT.

A given echo output entity is created for the purpose of providing the echoing function corresponding to measure values for a logical input device of some input class. Normally, the echo output entity would exist on a different CGI Virtual Device from the logical input device whose measure is being echoed. Functionally, echo output entities are concerned solely with output. The connection with the measure of any input device (or other source of value to be echoed) is the responsibility of the client of the CGI Virtual Device.

Echo output entities are only logical constructs; they need not correspond to any physical reality. Once an echo output has been set up (using INITIALIZE ECHO OUTPUT and ECHO OUTPUT CONTROLS), it is expected that new values for echoing will be repeatedly sent by the client using UPDATE <input class> ECHO OUTPUT.

The operating state of each echo output entity is specified in the corresponding Individual Echo Entity State List by the Echo Entity State entry.

The states in which it is not permissible to invoke the various echo output functions are defined in table 14, and in terms of the echo output entity state diagram, figure 5. This state diagram depicts transitions of the value of the Echo Entity State variable. The state transitions always take place as a result of the client invoking a function.

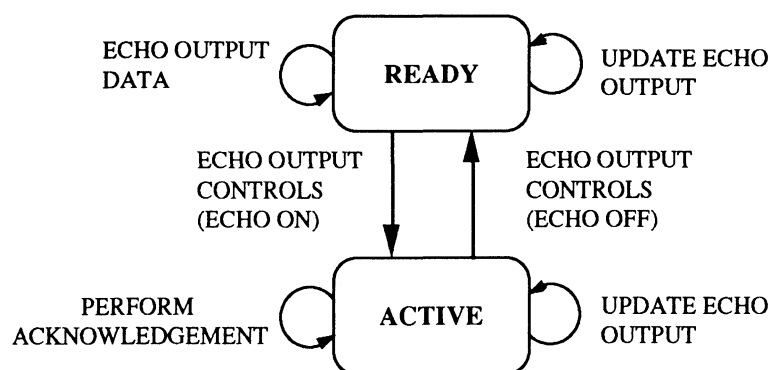


Figure 5 – Echo Output Entity State diagram

As an example of the use of remote echoing, consider a program which is simultaneously a client of two different CGI Virtual Devices: one an INPUT device, the other an OUTPUT device. Assume that there is a logical input device of type LOCATOR on the INPUT device. The client program should echo measure values from the INPUT device on the OUTPUT device. The following sequence of operations can occur:

- 1) The client uses PUT CURRENT LOCATOR MEASURE on the INPUT device to initialize the measure of the locator. (This step is optional.)
- 2) The client uses INITIALIZE ECHO OUTPUT to set the Echo Entity State to state READY.
- 3) The client uses ECHO OUTPUT DATA, if necessary, to customize the echo output entity.

- 4) The client uses SAMPLING STATE on the INPUT device to enable sampling.
- 5) The client uses SAMPLE LOCATOR on the INPUT device to obtain the locator measure.
- 6) The client uses SAMPLING STATE on the INPUT device to disable sampling.
- 7) The client sends the current value to the OUTPUT device using UPDATE LOCATOR ECHO OUTPUT.
- 8) On the INPUT device, the client uses INITIALIZE ECHO REQUEST to arm the locator's trigger(s) and start the timeout interval.
- 9) ECHO OUTPUT CONTROLS (LOCATOR, entity index, ECHO ON, PROMPT ON) is used on the OUTPUT device to perform the prompt action and to generate the echo of the current value; the Echo Entity State is set to state ACTIVE.
- 10) The client sends an ECHO REQUEST LOCATOR to the INPUT device. When the function returns control to the client, there are four cases depending on the value of the returned request status parameter:
  - MEASURE CHANGED* – The client passes the returned new value to the OUTPUT device using UPDATE LOCATOR ECHO OUTPUT and goes back to step 10.
  - TRIGGER FIRED* – The client notes the returned measure value and its validity.
  - BREAK* or *TIMEOUT* – The client notes that it has not received a valid input value.
- 11) If the returned request status parameter was TRIGGER FIRED then the client uses PERFORM ACKNOWLEDGEMENT to effect acknowledgement on the OUTPUT device.
- 12) The client uses ECHO OUTPUT CONTROLS (LOCATOR, entity index, ECHO OFF, PROMPT OFF) on the OUTPUT device to terminate the echoing process and set the Echo Entity State to state READY.
- 13) If it will not need the echo output entity again, the client releases those resources associated with it by using the RELEASE ECHO OUTPUT function on the OUTPUT device.

### 3.7.5 Event input

When the event input method is used, an operator may create input asynchronously by firing the triggers of one or more logical input devices which are enabled for event input. In response to such a trigger, an *event report* is normally stored in the *event queue* and an acknowledgement performed if Acknowledgement Control is ENABLED. The event queue is a first-in first-out queue that is common to all logical input devices and is part of the Virtual Device. The event report contains the logical input device class, the logical input device index, the trigger index of the trigger which caused the event, a timestamp, the measure value, and the measure validity status. The timestamp can be used to serialize events from different Virtual Devices which are used simultaneously. If one trigger is associated with two or more LIDs, its firing creates more than one event, which are marked by identical timestamps. The order in which such simultaneous events are placed in the event queue is implementation-dependent. If the input device does not have a true clock facility, the timestamp may be implemented as a count of trigger firing, otherwise timestamps are real numbers in units of seconds.

There are several mechanisms which control event input. These are discussed in terms of states (see figure 6) and control entries in the Class-Independent Logical Input Device State Lists and the Event Input State List.

In order to use event input, it is first necessary to initialize the event queue (by use of INITIALIZE EVENT QUEUE) and then to enable events on at least one LID (by use of ENABLE EVENTS).

The Event Queue State variable in the Event Input State List may have one of five values:

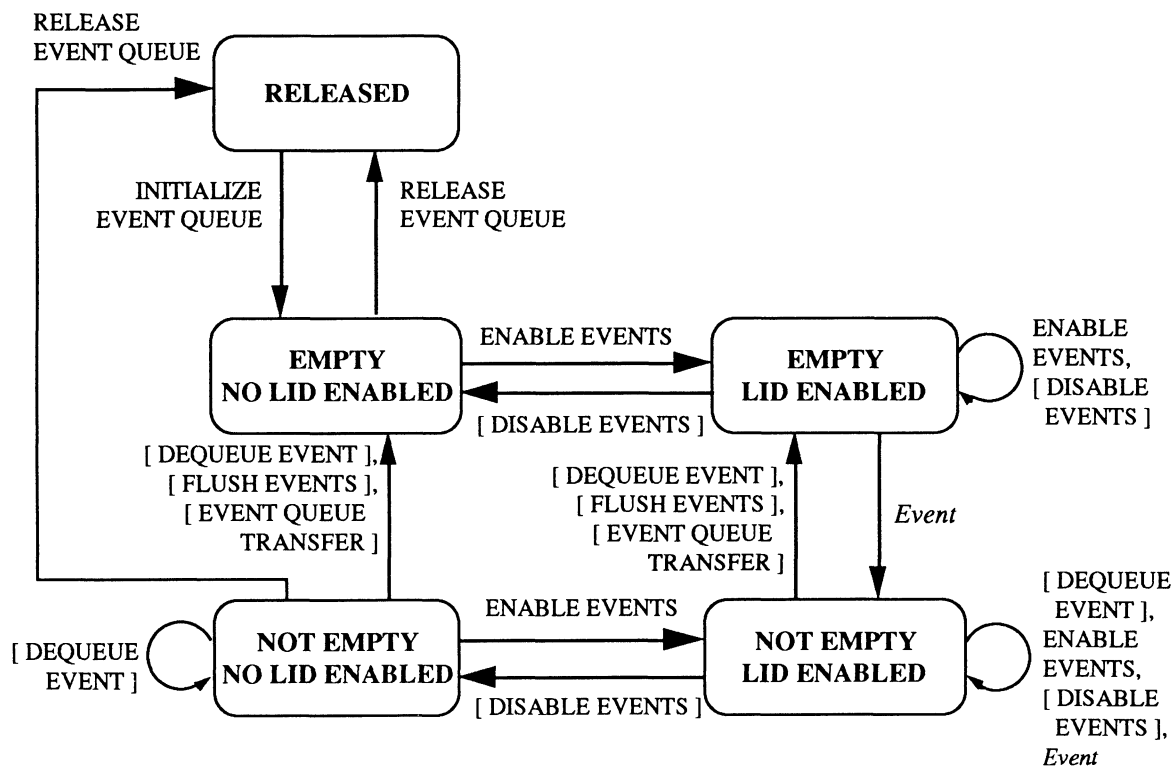
- RELEASED
- EMPTY NO LID ENABLED
- NOT EMPTY NO LID ENABLED
- EMPTY LID ENABLED
- NOT EMPTY LID ENABLED

This state variable restricts the usage of RELEASE EVENT QUEUE; it may only be released if there are no LIDs enabled for event input.

## Concepts

## Input methods and state model

If there is an attempt to trigger an event whose event report cannot be enqueued because of lack of space, the Unreported Overflow State and Event Queue Block State entries in the Event Input State List are set to OVERFLOW and BLOCKED respectively. The Unreported Overflow State entry is reset to NO OVERFLOW when it is reported to the client. If the Event Queue Block State is BLOCKED, no triggers for events are accepted and the echoes of the input devices which are enabled for events disappear. The Event Queue Block State may be set explicitly by the client, using the function EVENT QUEUE BLOCK CONTROL, if there are application dependent reasons for prohibiting entries in the event queue. The Event Queue Block State is reset explicitly by EVENT QUEUE BLOCK CONTROL.



[ ] indicates that a state transition is dependent on either the resulting number of LIDs enabled or the resulting number of events enqueued.

Figure 6 – Event Queue state diagram

The Unreported Break State entry in the Event Input State List is set to BREAK if the operator performs a break action. The break action does not create an event report entry in the event queue, the prompt is not affected, and the break action is not acknowledged. The part of the echo which reflected a partially constructed measure is removed. As with OVERFLOW, the Unreported Break State is reset to NO BREAK when it is reported to the client. Partially constructed string and stroke measures are discarded by the break action. Information about the most recent break action (if any) is held in the Event Input State List and is returned by AWAIT EVENT when its *event status* parameter has the value BREAK. If the break action is associated with a particular LID, identifying information for that LID is available; if the cause of the break action is defined generally for the CGI Virtual Device, the input class associated with the break is NONE and the LID index is undefined. The timestamp associated with any applicable break action is always defined.

For retrieving event input, the functions AWAIT EVENT, EVENT QUEUE TRANSFER, and DEQUEUE <input class> EVENT are provided.

The AWAIT EVENT function returns the event status, and (depending on the event status) the input class, the input device index, the trigger index, and the timestamp. The event status is set to EVENT if at least one event report is available in the event queue. The other parameters returned by AWAIT EVENT describe the oldest event in the queue.

TIMEOUT is returned as the event status if no event is available within the specified timeout interval and a break action was not performed. The response validity is returned as INVALID if an error condition exists, e.g. the event queue is not initialized, or AWAIT EVENT was invoked when the event queue is empty and blocked.

After an overflow occurs, the next invocation of AWAIT EVENT reports OVERFLOW, and subsequent calls indicate EVENT as long as event reports are available.

If the state of the event queue is both BREAK and OVERFLOW at the time AWAIT EVENT is invoked, BREAK is reported before OVERFLOW.

If the event status EVENT is returned by AWAIT EVENT, the client may call EVENT QUEUE TRANSFER or the appropriate DEQUEUE <input class> EVENT function to retrieve the required event report data. The function DEQUEUE <input class> EVENT removes one event report and passes the measure and measure validity status of this event to the client. The function EVENT QUEUE TRANSFER returns a data record containing an event reports list.

To clear the whole event queue or to delete events which belong to a certain device the functions FLUSH EVENTS or FLUSH DEVICE EVENTS may be used.

Table 2 defines which parameters of AWAIT EVENT, besides the event status, are valid for the different values of the event status.

Table 2 – Event Status of AWAIT EVENT

| Event Status | Valid Return Parameters                           |
|--------------|---------------------------------------------------|
| TIMEOUT      | none                                              |
| EVENT        | all                                               |
| OVERFLOW     | all                                               |
| BREAK        | timestamp, class, and LID index when class ≠ NONE |

### 3.8 Prompting, echoing, and acknowledgement

There are three mechanisms defined in CGI which control the interaction between the operator and the Virtual Device: *prompting* is a signal which tells the operator to start some input action; *echoing* is the visualization of the current value of a LID's measure; *acknowledgement* is the signal that the input action was accepted by the input device.

The form of prompt, echo, and acknowledgement is controlled for each logical input device by the respective Prompt Type, Echo Type, and Acknowledgement Type as specified in the Class-Independent Logical Input Device State List. For all logical input devices, prompt, echo, and acknowledgement type 1 is required to be supported and is implementation-dependent. Further prompt, echo, and acknowledgement types appropriate to each input class are defined but not required. These types are listed in the following tables. Other positive prompt, echo, and acknowledgement types are reserved for registration (see ISO/IEC 9636-1, 4.4.7); negative values for prompt, echo, and acknowledgement types are available for private use.

Additional information to control prompt, echo, and acknowledgement is passed in a data record, which the client may set through the <input class> DEVICE DATA function. In addition, prompt, echo, and acknowledgement behaviour of a LID may be customized in an implementation-dependent manner by use of the input device data record.

A Virtual Device of class OUTIN or OUTPUT with echo output capabilities shall support at least echo type 1 for each input class supported.

The Class-Independent Logical Input Device State List contains additional information for the control of prompting, echoing, and acknowledgement on an OUTIN Virtual Device. The Echo Area is a rectangular region specified by two VPs which may be used to position or restrict certain types of static non-geometric prompting, echoing, and acknowledgement. The Echo Data Record has contents which depend on the current Prompt Type, Echo Type, and Acknowledgement Type.

The functions ECHO CONTROLS and ECHO DATA are used to control the prompt, echo, and acknowledgement behaviour of the logical input device. See also 3.7.4.

#### 3.8.1 Prompting

Prompting is a form of output (possibly non-graphical) which indicates that the LID is ready for input and which occurs when the Prompt Control is ENABLED and the LID becomes active. The concept of an *active* LID is described in 3.7. Examples of prompting are displaying a prompt message on the display surface of an OUTIN Virtual Device or sounding a tone on an INPUT Virtual Device. Conceptually, drawing surfaces and particularly bitmaps (see ISO/IEC 9636-6) are unaffected by prompting.

Table 3 – Prompt Types

| Number | Description                                                                                                                                                                                                           |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1      | Implementation-dependent. No data record is required in the echo data record.                                                                                                                                         |
| 2      | Prompt by sounding <i>a tone</i> . For devices capable of sounding multiple pitches, a prompt tone is given in the echo data record. The echo data record contains the following prompt information:<br>Prompt tone I |
| 3      | Prompt by displaying one from a set of <i>device prompt outputs</i> , such as LED indicators or icons. The echo data record contains the following prompt information:<br>Prompt display identifier IX                |
| 4      | Display a <i>prompt message</i> . The echo data record contains the following prompt information:<br>Prompt message S                                                                                                 |

### 3.8.2 Echoing

The current value of a logical input device's measure may be made visible to the operator via the process of echoing. While the logical input device is active and the Echo Control is ENABLED, the current measure value is displayed in a form determined by the logical input device's Echo Type. For example, one echo type for a LOCATOR input device may represent the current measure value by the position of crosshairs on a display surface, while another type might display the numeric coordinates of the current measure value near a cursor. Conceptually, drawing surfaces and particularly bitmaps (see ISO/IEC 9636-6) are unaffected by echoing.

For an OUTIN Virtual Device, CGI functions are defined that control the use of its display surface to echo the measure values of its logical input devices; these functions also apply to an INPUT Virtual Device, to the extent that the echoing method need not employ graphical output (for example, a digital numeric display).

Table 4 – Echo Types: LOCATOR

| Number | Description                                                                                                                                                                                                                                                                         |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1      | Implementation-dependent. No data record is required in the echo data record.                                                                                                                                                                                                       |
| 2      | Echo the current value with a <i>crosshair</i> (i.e. horizontal and vertical lines spanning the display surface and intersecting at the current value position).                                                                                                                    |
| 3      | Echo the current value with a <i>tracking symbol</i> . The echo data record contains the following echo information:<br>Symbol identifier IX                                                                                                                                        |
| 4      | Echo the current value with a <i>rubber band line</i> . The starting point of the line is given in the echo data record; the end point is the current value position. The echo data record contains the following echo information:<br>Initial position P                           |
| 5      | Echo the current value with a <i>rubber band rectangle</i> . The starting point of the rectangle's diagonal line is given in the echo data record; the end point is the current value position. The echo data record contains the following echo information:<br>Initial position P |
| 6      | Display a <i>digital representation</i> of the current value within the echo area.                                                                                                                                                                                                  |

Table 5 – Echo Types: STROKE

| Number | Description                                                                                                                                                                                                                                                                                                               |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1      | Implementation-dependent. No data record is required in the echo data record.                                                                                                                                                                                                                                             |
| 2      | Display a <i>digital representation</i> of the current stroke within the echo area                                                                                                                                                                                                                                        |
| 3      | Display a <i>symbol</i> at each stroke point and a <i>tracking symbol</i> at the current stroke point. The echo data record contains the following echo information:<br><div style="text-align: right;">Symbol identifier IX<br/>Tracking symbol identifier IX</div>                                                      |
| 4      | Display a <i>line</i> joining each stroke point and a <i>rubber band line</i> between the current position and the previous point.                                                                                                                                                                                        |
| 5      | Echo the current value with a <i>rubber band rectangle</i> . The starting point of the rectangle's diagonal line is given in the echo data record; the end point is the current value position. The echo data record contains the following echo information:<br><div style="text-align: right;">Initial position P</div> |
| 6      | Display a <i>symbol</i> at each stroke point and at the current value position display a crosshair (i.e. horizontal and vertical lines spanning the display surface and intersecting at the current value position).<br><div style="text-align: right;">Symbol identifier IX</div>                                        |

Table 6 – Echo Types: CHOICE

| Number | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1      | Implementation-dependent. No data record is required in the echo data record.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 2      | Display <i>CHOICE strings</i> and alter the appearance of the currently selected string. The choice strings are contained in the echo data record and are displayed within the echo area. The operator may choose among the strings using an appropriate technique. The echo data record contains the following echo information:<br><div style="text-align: right;">List of strings nS</div>                                                                                                                                                                                                                                                                                                  |
| 3      | Display <i>segment</i> and alter the appearance of the primitives for the currently selected pick identifier. The echo data record contains a segment identifier for a segment to be displayed within the echo area. The segment transformation is applied, the default VDC extent for the currently selected VDC type is mapped to the echo area. The active VDC extent is ignored. The pick identifiers in the echo data record are mapped to CHOICE numbers. Picking these primitives selects the corresponding CHOICE value. The echo data record contains the following echo information:<br><div style="text-align: right;">Segment identifier SN<br/>List of pick identifiers nPN</div> |

Table 7 – Echo Types: VALUATOR

| Number | Description                                                                          |
|--------|--------------------------------------------------------------------------------------|
| 1      | Implementation-dependent. No data record is required in the echo data record.        |
| 2      | Display a <i>graphical representation</i> of the current value within the echo area. |
| 3      | Display a <i>digital representation</i> of the current value within the echo area.   |

Table 8 – Echo Types: PICK

| Number | Description                                                                                                                                                                                                                                             |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1      | Implementation-dependent. No data record is required in the echo data record.                                                                                                                                                                           |
| 2      | <i>Alter the appearance</i> of all objects with a pick value that would be returned by the input operation.                                                                                                                                             |
| 3      | <i>Alter the appearance</i> of all objects within the segment that would be picked.                                                                                                                                                                     |
| 4      | Display the current VDC position with a <i>tracking symbol</i> . The echo data record contains the following echo information:<br>Symbol identifier IX                                                                                                  |
| 5      | <i>Alter the appearance</i> of all objects with a pick value that would be returned by the input operation and display the position of the cursor using a <i>tracking symbol</i> ; the echo data record contains the following:<br>Symbol identifier IX |
| 6      | <i>Alter the appearance</i> of all objects within the segment that would be picked and display the position of the cursor using a <i>tracking symbol</i> ; the echo data record contains the following:<br>Symbol identifier IX                         |

Table 9 – Echo Types: STRING

| Number | Description                                                                                                                                                                                                                                               |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1      | Implementation-dependent. No data record is required in the echo data record.                                                                                                                                                                             |
| 2      | Display the input character string in the echo area using code extension techniques defined by Input Character Coding Announcer, Input Character Set Index and Alternate Input Character Set Index in the Class-Specific Logical Input Device State List. |

Table 10 – Echo Types: RASTER

| Number | Description                                                                          |
|--------|--------------------------------------------------------------------------------------|
| 1      | Implementation-dependent. No data record is required in the echo data record.        |
| 2      | Display a <i>graphical representation</i> of the current value within the echo area. |

Table 11 – Echo Types: GENERAL

| Number | Description                                                                   |
|--------|-------------------------------------------------------------------------------|
| 1      | Implementation-dependent. No data record is required in the echo data record. |

### 3.8.3 Acknowledgement

Acknowledgement is performed if the Acknowledgement Control is ENABLED, a trigger has fired, and the measure has been accepted. The form of acknowledgement is specified by the Acknowledgement Type. Conceptually, drawing surfaces and particularly bitmaps (see ISO/IEC 9636-6) are unaffected by acknowledgements.

A break action performs no acknowledgement.

Table 12 – Acknowledgement Types

| Number | Description                                                                                                                                                                                                                                            |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1      | Implementation-dependent. No data record is required in the echo data record.                                                                                                                                                                          |
| 2      | Acknowledge by <i>sounding a tone</i> . For devices capable of sounding multiple pitches, an acknowledgement tone is given in the echo data record. The echo data record contains the following acknowledgement information:<br>Acknowledgement tone I |
| 3      | Display an <i>acknowledgement message</i> . The echo data record contains the following acknowledgement information:<br>Acknowledgement message S                                                                                                      |
| 4      | The colour of the displayed prompt changes.                                                                                                                                                                                                            |

### 3.9 Portioning of returned input data

For STROKE, PICK, STRING, and RASTER, input devices, the returned value is a list of data. If this data cannot be returned in one portion with the related functions REQUEST <input class>, ECHO REQUEST <input class>, SAMPLE <input class>, or DEQUEUE <input class> EVENT, then the GET ADDITIONAL <input class> DATA functions may be used. These functions have to be invoked before the next input function for the same LID, otherwise, the data is lost.

The GET ADDITIONAL <input class> DATA function is only used to portion data at the interface to the client and returns one portion of the data representing the complete measure of the corresponding previous REQUEST, ECHO REQUEST, SAMPLE, or DEQUEUE function. It is independent of the effective buffer size for the various types of input data given in the Class-Specific Logical Input Device State List.

### 3.10 State restrictions

The state variables, Input Device State and Sampling State, defined in the Class-Independent Logical Input Device State List, and the Event Queue State defined in the Event Input State List provide certain state restrictions over the functions defined in this part of ISO/IEC 9636.

Tables 13, 14 and 15 list the state restrictions that apply. Functions not listed in these tables are not restricted.

Table 13 – State Restrictions For Input Functions

| Function                          | Input Device State | Sample State | Event Queue State |
|-----------------------------------|--------------------|--------------|-------------------|
| INITIALIZE LOGICAL INPUT DEVICE   | -                  | -            | -                 |
| RELEASE LOGICAL INPUT DEVICE      | -                  | -            | -                 |
| ECHO CONTROLS                     | RL, EV, ERE, ERC   | -            | -                 |
| PUT CURRENT <input class> MEASURE | RL                 | -            | -                 |
| ECHO DATA                         | RL, EV, ERE, ERC   | -            | -                 |
| <input class> DEVICE DATA         | RL, EV, ERE, ERC   | -            | -                 |
| ASSOCIATE TRIGGERS                | RL                 | -            | -                 |
| REQUEST <input class>             | RL, ERE, ERC, EV   | -            | -                 |
| SAMPLE <input class>              | RL                 | SD           | -                 |
| SAMPLING STATE                    | RL                 | -            | -                 |
| INITIALIZE ECHO REQUEST           | RL, ERE, ERC, EV   | -            | -                 |
| ECHO REQUEST <input class>        | RL, RY, EV         | -            | -                 |

Table 13 – State Restrictions For Input Functions – (concluded)

| Function                    | Input Device State | Sample State | Event Queue State |
|-----------------------------|--------------------|--------------|-------------------|
| RELEASE EVENT QUEUE         | -                  | -            | ELE, NELE         |
| ENABLE EVENTS               | RL, ERE, ERC, EV   | -            | RL                |
| DISABLE EVENTS              | RL                 | -            | -                 |
| EVENT QUEUE BLOCK CONTROL   | -                  | -            | RL                |
| FLUSH EVENTS                | -                  | -            | RL                |
| FLUSH DEVICE EVENTS         | -                  | -            | RL                |
| AWAIT EVENT                 | -                  | -            | RL, ENLE          |
| DEQUEUE <input class> EVENT | -                  | -            | RL, ENLE, ELE     |
| EVENT QUEUE TRANSFER        | -                  | -            | RL, ENLE, ELE     |

Table 14 – State Restrictions For Echo Output Functions

| Echo Output Entity State         |                   |
|----------------------------------|-------------------|
| Function                         | Echo Entity State |
| INITIALIZE ECHO OUTPUT           | -                 |
| RELEASE ECHO OUTPUT              | -                 |
| ECHO OUTPUT CONTROLS             | -                 |
| PERFORM ACKNOWLEDGEMENT          | RY                |
| UPDATE <input class> ECHO OUTPUT | -                 |
| ECHO OUTPUT DATA                 | AC                |

Table 15 – Key to abbreviations in tables 13 and 14.

| Input Device States      | Abbrev. | Sampling States    | Abbrev. |
|--------------------------|---------|--------------------|---------|
| RELEASED                 | RL      | DISABLED           | SD      |
| READY                    | RY      |                    |         |
| ECHO REQUEST ENABLED     | ERE     |                    |         |
| ECHO REQUEST COMPLETED   | ERC     |                    |         |
| EVENTS ENABLED           | EV      |                    |         |
| Event Queue States       | Abbrev. | Echo Entity States | Abbrev. |
| RELEASED                 | RL      | READY              | RY      |
| EMPTY LID ENABLED        | ELE     | ACTIVE             | AC      |
| EMPTY NO LID ENABLED     | ENLE    |                    |         |
| NOT EMPTY NO LID ENABLED | NENL    |                    |         |
| NOT EMPTY LID ENABLED    | NELE    |                    |         |

### 3.11 Inquiry

Input inquiry functions, as defined in clause 6, provide the client with the means to access the information in the Input and Echoing state lists and description tables. These lists and tables provide information about the current state and capabilities of the CGI Virtual Device.

Details about the relationship between a description table or state list and the corresponding inquiry functions are described in ISO/IEC 9636-1.

## 4 Interactions with other parts of ISO/IEC 9636

### 4.1 Interactions with more than one part of ISO/IEC 9636

The input states defined in this part of ISO/IEC 9636 are orthogonal to states defined in other parts of ISO/IEC 9636.

### 4.2 Interactions with ISO/IEC 9636-1 (Overview)

The Principles of Inquiry given in ISO/IEC 9636-1 require that functions which return information provide an indication whether that information is valid or invalid. For the functions defined in this part of ISO/IEC 9636, there is a response validity flag with values INVALID or VALID. When an error reaction states, "Function ignored", this means that the return parameter(s) are set to the appropriate values to indicate an error resulting in invalid data, but that the function returns immediately with no other effect.

### 4.3 Interactions with ISO/IEC 9636-2 (Control)

The INITIALIZE function will release all logical input devices, all echo output entities, and any event queue. There are no state restrictions on the use of INITIALIZE and TERMINATE, i.e. INITIALIZE and TERMINATE may be used at any time (see ISO/IEC 9636-2, 5.2.1).

The default for the Input Extent is obtained from the output VDC Extent current at the time of INITIALIZE LOGICAL INPUT DEVICE. Changing the VDC type sets the Input Extent to the default for the chosen type. The default for the Input Viewport is based upon the Device Viewport current at the time of INITIALIZE LOGICAL INPUT DEVICE. The method of calculation is described in clause 3.

Input operations for class PICK are affected by the current VDC-to-Device Mapping.

### 4.4 Interactions with ISO/IEC 9636-3 (Output)

There are no interactions between this part of ISO/IEC 9636 and the functions and features specified in ISO/IEC 9636-3. However, an implementation may wish to make use of some of the attributes specified for output when defining implementation-dependent echoes.

### 4.5 Interactions with ISO/IEC 9636-4 (Segments)

Input operations for class PICK return pick values which consist of segment name and pick name created with functions defined in ISO/IEC 9636-4. The segment attributes of Detectability, Visibility, and Pick Priority determine which segments are candidates for picking.

The function GET ADDITIONAL PICK DATA, defined in this part of ISO/IEC 9636, may be enabled following execution of the SIMULATE PICK function, defined in ISO/IEC 9636-4.

The states specified in this part of ISO/IEC 9636 are orthogonal to those specified in ISO/IEC 9636-4, i.e. the two sets of states do not interact and any combination of them is allowed.

## 4.6 Interactions with ISO/IEC 9636-6 (Raster)

There are no interactions between this part of ISO/IEC 9636 and ISO/IEC 9636-6. In particular, output of prompt, echo, and acknowledgement is conceptually performed downstream of any drawing or display bitmap and shall not produce client detectable effects in these bitmaps.

## 5 Abstract specification of functions

### 5.1 Introduction

This clause describes the abstract functional capabilities of the Input and Echoing part of the CGI.

To avoid replicating the description of certain groups of related functions, a generic description technique is employed using “<input class>” in the function name. It should be emphasized that these related functions will not be represented by a single function in a binding or encoding of the CGI. For functions in which none of the parameter data types depend on the input class, there is only one function for all classes and the input class is passed as one explicit *In* parameter of the function.

The descriptions of functions are ordered in the following manner:

- Input control functions.
- Input functions, which provide for the return of graphical and non-graphical operator-set data values in request or sample modes;
- Echo request input functions.
- Input functions associated with event mode input;
- Echo output functions.

#### 5.1.1 Data types employed

The abstract specifications of functions detail the functions in terms of input and output parameters. The data type of each parameter is selected from a standard set and is identified in the functional specification by a standard abbreviation.

The data types and the abbreviations used in this part of ISO/IEC 9636 are taken from the complete list of data types given in ISO/IEC 9636-1, 5.2.10.

#### 5.1.2 Validity of returned information

For all of the functions specified in this clause which solicit a response from the Virtual Device, a response validity flag is returned as INVALID if an error was detected in executing the function. In such cases, other output parameters are undefined and no meaning should be applied to any of these parameter values. Additionally, functions which return measure values also return an associated measure validity flag which applies only to the measure value itself. It is possible to get a response in which the response validity is VALID and the measure validity is INVALID.

## 5.2 Input control functions

### 5.2.1 INITIALIZE LOGICAL INPUT DEVICE

#### Parameters:

|           |                    |                                                                       |    |
|-----------|--------------------|-----------------------------------------------------------------------|----|
| <i>In</i> | input class        | (LOCATOR, STROKE, VALUATOR, CHOICE,<br>PICK, STRING, RASTER, GENERAL) | E  |
| <i>In</i> | input device index | (1..n)                                                                | IX |

#### Effect:

The Input Device State of the specified logical input device becomes READY. Any input actions are terminated. The Logical Input Device State List is set to its default initial values.

**Abstract specification of functions****Input control functions**

NOTE – If the logical input device is not in the state RELEASED, then INITIALIZE LOGICAL INPUT DEVICE is functionally equivalent to RELEASE LOGICAL INPUT DEVICE followed by INITIALIZE LOGICAL INPUT DEVICE. Hence, this function may be used in any LID state to return the logical input device to its default state.

**State Restrictions:**

This function may be invoked in all LID states.

**5.2.2 RELEASE LOGICAL INPUT DEVICE****Parameters:**

|           |                    |                                                                       |    |
|-----------|--------------------|-----------------------------------------------------------------------|----|
| <i>In</i> | input class        | (LOCATOR, STROKE, VALUATOR, CHOICE,<br>PICK, STRING, RASTER, GENERAL) | E  |
| <i>In</i> | input device index | (1..n)                                                                | IX |

**Effect:**

The Input Device State of the specified logical input device becomes RELEASED. Any input actions are terminated.

**State Restrictions:**

This function may be invoked in all LID states.

**5.2.3 ECHO CONTROLS****Parameters:**

|           |                         |                                                                       |    |
|-----------|-------------------------|-----------------------------------------------------------------------|----|
| <i>In</i> | input class             | (LOCATOR, STROKE, VALUATOR, CHOICE,<br>PICK, STRING, RASTER, GENERAL) | E  |
| <i>In</i> | input device index      | (1..n)                                                                | IX |
| <i>In</i> | prompt control          | (DISABLED, ENABLED)                                                   | E  |
| <i>In</i> | echo control            | (DISABLED, ENABLED)                                                   | E  |
| <i>In</i> | acknowledgement control | (DISABLED, ENABLED)                                                   | E  |

**Effect:**

The Prompt Control, Echo Control, and Acknowledgement Control entries in the Class-Independent Logical Device State List of the specified logical input device are set to the specified value.

**State Restrictions:**

This function may only be invoked in LID State READY.

**Errors:**

|                          |                                                      |
|--------------------------|------------------------------------------------------|
| <i>Error identifier:</i> | 5:501                                                |
| <i>Cause:</i>            | Function illegal in LID State RELEASED               |
| <i>Reaction:</i>         | Function ignored.                                    |
| <i>Error identifier:</i> | 5:502                                                |
| <i>Cause:</i>            | Function illegal in LID State ECHO REQUEST ENABLED   |
| <i>Reaction:</i>         | Function ignored.                                    |
| <i>Error identifier:</i> | 5:503                                                |
| <i>Cause:</i>            | Function illegal in LID State ECHO REQUEST COMPLETED |
| <i>Reaction:</i>         | Function ignored.                                    |
| <i>Error identifier:</i> | 5:504                                                |
| <i>Cause:</i>            | Function illegal in LID State EVENTS ENABLED         |
| <i>Reaction:</i>         | Function ignored.                                    |
| <i>Error identifier:</i> | 5:514                                                |
| <i>Cause:</i>            | Function illegal in LID Sampling State ENABLED       |
| <i>Reaction:</i>         | Function ignored.                                    |

## Input control functions

## Abstract specification of functions

## 5.2.4 PUT CURRENT &lt;input class&gt; MEASURE

## Parameters:

|           |                            |                             |           |
|-----------|----------------------------|-----------------------------|-----------|
| <i>In</i> | input device index         | (1..n)                      | IX        |
| <i>In</i> | measure validity           | (INVALID, VALID)            | E         |
| <i>In</i> | measure value              |                             |           |
|           | if input class = LOCATOR:  | position                    | P         |
|           | if input class = STROKE:   | list of points              | nP        |
|           | if input class = VALUATOR: | value                       | R         |
|           | if input class = CHOICE:   | choice number               | (1..n) I  |
|           | if input class = PICK:     | list of pick values         | nPV       |
|           | if input class = STRING:   | string                      | S         |
|           | if input class = RASTER:   | xcount, ycount              | (1..n) 2I |
|           |                            | list of input colour values | nICO      |
|           | if input class = GENERAL:  | data record                 | D         |

## Effect:

The measure value and the measure validity are set for the specified logical input device. The resulting measure validity is VALID if the *measure value* parameter is valid for the measure of the specified logical input device and the *measure validity* parameter is VALID. In all other cases, the resulting measure validity shall be INVALID. When a specified *measure validity* of VALID is overridden by an invalid measure value, the situation is not regarded as an error.

Some input devices will not be able to set the measure; in this case, the function is ignored. The Put Current Measure Effective entry in the Class-Independent Logical Input Device State List indicates whether this function has any effect on the LID.

NOTE – This function may be used to set the initial value before an input action is performed. It can be used for repositioning the built-in echo on an OUTIN device.

## State Restrictions:

This function may be invoked in all LID states except state RELEASED.

## Errors:

|                          |                                                      |
|--------------------------|------------------------------------------------------|
| <i>Error identifier:</i> | 3:504                                                |
| <i>Cause:</i>            | PUT CURRENT MEASURE not effective for this LID       |
| <i>Reaction:</i>         | Function ignored.                                    |
| <i>Error identifier:</i> | 3:508                                                |
| <i>Cause:</i>            | Too many stroke points for this LID                  |
| <i>Reaction:</i>         | Maximum stroke buffer size used.                     |
| <i>Error identifier:</i> | 3:509                                                |
| <i>Cause:</i>            | Choice number exceeds maximum available for this LID |
| <i>Reaction:</i>         | Function ignored.                                    |
| <i>Error identifier:</i> | 3:510                                                |
| <i>Cause:</i>            | Too many pick values for this LID                    |
| <i>Reaction:</i>         | Maximum pick buffer used.                            |
| <i>Error identifier:</i> | 3:511                                                |
| <i>Cause:</i>            | Too many string characters for this LID              |
| <i>Reaction:</i>         | Maximum string buffer used.                          |
| <i>Error identifier:</i> | 3:513                                                |
| <i>Cause:</i>            | General input data record too large                  |
| <i>Reaction:</i>         | Maximum data record size used.                       |
| <i>Error identifier:</i> | 3:521                                                |
| <i>Cause:</i>            | Valuator value out of range                          |
| <i>Reaction:</i>         | Function ignored.                                    |