ISO/IEC JTC 1/SC 35 N1052-E

DATE: 2006-10-05

ISO/IEC JTC 1/SC 35

User Interfaces

Secretariat: AFNOR

DOCUMENT TYPE: Working document

TITLE: ISO/IEC 24757 Information Technology — Keyboard Interaction model — Machine readable keyboard description

SOURCE: JTC1/SC35 WG1

ACTION Identifier : For action

DUE DATE : 2007-02-04

DISTRIBUTION: P members

MEDIUM: E

NO. OF PAGES: 88

Secretariat ISO/IEC JTC 1/SC 35 : AFNOR – Nathalie Cappel-Souquet
Address : 11 rue Francis de Pressensé - 93571 La Plaine Saint-Denis Cedex - France
Telephone: +33 1 41 62 82 55; Facsimile:+33 1 49 17 90 00
E-mail nathalie.cappelsouquet@afnor.org

FCD ISO/IEC 24757

Final Committee Draft ISO/IEC FCD 24757	
Date: **2006-10-05**	Reference number: ISO/IEC JTC 1/SC 35 **N 1052-E & 1052-F**
Supersedes document SC YY N XXXX	

THIS DOCUMENT IS STILL UNDER STUDY AND SUBJECT TO CHANGE. IT SHOULD NOT BE USED FOR REFERENCE PURPOSES.

ISO/IEC JTC 1/SC 35 User Interfaces Secretariat: AFNOR	Circulated to P- and O-members, and to technical committees and organisations in liaison for voting (P-members only) by: **2007-02-04** Please return all votes and comments in electronic form directly to the SC 35 Secretariat by the due date indicated.

ISO/IEC 24757

Title: Information Technology — Keyboard Interaction model — Machine-readable keyboard description / Modèle d'interaction des claviers -Description lisible à la machine des interactions sur clavier

Project: 1.xx.xx.xx.xx

Introductory note:

Medium:E

No. of pages:

Address Reply to: Secretariat, ISO/IEC JTC 1/SC 35, Address
AFNOR –Nathalie Cappel-Souquet Address : 11 rue Francis de Pressensé - 93571 La Plaine Saint-Denis Cedex - France Telephone: +33 1 41 62 82 55; Facsimile: +33 1 49 17 90 00 E-mail: nathalie.cappelsouquet@afnor.org

© ISO/IEC 2005 – All rights reserved

ISO/IEC JTC 1/SC 35 N 1052 E

Date: 2005-06-13

ISO/IEC WD 24757

ISO/IEC JTC 1/SC 35/WG 1

Secretariat: AFNOR

Information Technology — Keyboard Interaction model — Machine-readable keyboard description

Technologies de l'information — Intercations sur clavier — Modèler

Warning

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: International Standard
Document subtype:
Document stage: (20) Preparatory
Document language: E

C:\Documents and Settings\ncs\Mes documents\z62f\jeju\35n1052E-FCD-24757-20060908.doc~~D:\Donnees\SC35\ISO 24757 Description interactions clavier\ISO IEC FCD 24757 E-20060905.doc~~ STD Version 2.1c2

Copyright notice

This ISO document is a working draft or committee draft and is copyright-protected by ISO. While the reproduction of working drafts or committee drafts in any form for use by participants in the ISO standards development process is permitted without prior permission from ISO, neither this document nor any extract from it may be reproduced, stored or transmitted in any form for any other purpose without prior written permission from ISO.

Requests for permission to reproduce this document for the purpose of selling it should be addressed as shown below or to ISO's member body in the country of the requester:

[Indicate the full address, telephone number, fax number, telex number, and electronic mail address, as appropriate, of the Copyright Manger of the ISO member body responsible for the secretariat of the TC or SC within the framework of which the working document has been prepared.]

Reproduction for sales purposes may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

© ISO/IEC 2005 – All rights reserved

Contents

Page

© ISO/IEC 2005 – All rights reserved

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. 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.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 35, *User interfaces*.

© ISO/IEC 2005 – All rights reserved

Introduction

This International Standard is destined to hose who design operating systems or software applications taking the keyboard in charge (including for a complete presentation of the keyboard on the screen for documentation ends). Its goal is to harmonize industry practices (PCs, PDAs, Linux, Windows, Apple, …). It finally aims at easing the production of drivers interoperable for the user and to allow a better assistance to the user by offering a more precise mapping between the physical keyboard as engraved and geometrically configured, and the logical interface of which are aware the operating system and its applications.

© ISO/IEC 2005 – All rights reserved

Inmformation Technology — Keyboard Interaction — ModelInmformation Technology — Keyboard Interaction — Model

1 Scope

This International Standard aims at providing a formal description format that can not only describe the international keyboards standards fully, but also fully describe the capabilities of keyboards in the marketplace of today and the foreseeable future and their functioning with corresponding operating systems. It describes possible interactions between the keys of a keyboard and standardizes the keyboard description so that it be machine-readable while staying relatively easy to interpret by human beings.

2 Conformance

The machine-readable description of a keyboard is conformant to this International Standard if it meets the requirements of clauses 5.x to 5.y.

3 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this International Standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies (included amendments).

ISO 639-2:1998, *Codes for the representation of names of languages -- Part 2: Alpha-3 code*

ISO 3166-1:1997, *Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes*

ISO/IEC 9995-1:2006, *Information technology – Keyboard layouts for text and office systems – Part 1: General principles governing keyboard layouts*

ISO/IEC 10646:2003*, Information technology –Universal Multiple-Octet Coded Character Set (UCS)*

ISO/IEC 19757-2:2003 *Information technology -- Document Schema Definition Language (DSDL) -- Part 2: Regular-grammar-based validation -- RELAX NG*

4 Terms and definitions

For the purposes of ISO/IEC 24757, the terms and definitions given in ISO/IEC 9995-1 apply. The following definitions also apply.

© ISO/IEC 2005 – All rights reserved

4.1
reference coordinates
identifier formed by one letter and two decimal digits, refering to the numbering grid allowing to position a key at the intersection of a row and a column of the keyboard, where rows are identified by letters and columns by digits

5 Requirements

5.1 Description of keys and keyboard layouts

The keyboard is described for machines in this International Standard using the following properties:

- the logical coordinates of each key according to a grid numbering system established in ISO/IEC 9995-1. These coordinates are called “reference coordinates”;

  Note: The reference coordinates of a keyboard may refer to a keyboard standard.

- the description of the size in millimeters (mm), shape and physical coordinates of each key of a hardware keyboard, to allow reproducing the user’s keyboard precisely on a computer screen; a binding is made with the reference coordinates;

- optionally, principal region or country and language codes applicable to this layout;

- actual labeling appearance;

- the logical division of each key of the keyboard into groups and levels within each group.

Within a group, at a given level, each character is described using five other properties:

- by an identifier formed with the letter U and the number it is assigned in the Universal character set (ISO/IEC 10646); if the character is a composed or combined character, identifiers of the individual characters making up the resulting character are given with the sequence that allows this composition ;

- optionally, by its character name in a given natural language, identified according to ISO 639-1 code or ISO 639-2 terminology code;

- optionally, by one or many "scan codes" (which may vary depending on the keyboard state, on the hardware in use, etc.);

- in case a key position makes a key become a dead key, a description of user-required combinations, with resulting character or characters.

- optionally, a tag to identify characters whose glyph engraved on the key top is different from its internal representation (e.g., U+00A6 broken vertical bar vs. U+007C solid vertical bar)

Note: The description of required combinations does not preclude the keyboard drivers to generate other characters if these combinations are already defined. The description mentioned here makes sure that what the end-user needs is taken care of, regardless of previous definitions.

This description includes the interaction of keys between them to produce results.

The following clause describes the different states of a keyboard necessary to the input of characters and taken into account in this International Standard.

© ISO/IEC 2005 – All rights reserved

5.2 States resulting from the interaction of function keys used for input

Hitting one of the following function keys determines a state of the keyboard that produces the desired character. This clause standardizes the conventional states produced by the interaction of these keys with alphanumeric keys.

- Level 2 Select (also called « Shift »): per se, this key allows input of level 2 characters in the active group.

- Level 3 Select (also called « AltGr »): per se, this key allows input of level 3 characters in the active group.

- Control : this key is often used in practice with one of the preceding keys for entering characters. See below the interpretation that this International Standard prescribes for these interactions.

- Group Select: per se, this key allows changing group. This action may be locking or not.

  (This description in an annex and – to do: generalize group selection)

  ISO/IEC 9995-2 specifies what follows to this effect :

  For the input of graphic character repertoire of collection 281 (titled MES-1) as specified in amendment 1 to ISO/IEC 10646:1-2000, a *Common Secondary Group Layout* (to be used as group 2) is specified in ISO/IEC 9995-3. Specifically for group 2, the activation of group 2 with the *Group select* function is recommended to be latching for the next character entered and for this character only. In other words, activation of group 2 changes the logical state of the keyboard so that all keys involved in this activation can be released, and still, the next key typed will be selecting a character in group 2. After typing such a character in this mode, the keyboard then reverts back automatically to the group active before group 2 wast activated.

  NOTE. It is recommended, when a group which defines a complete script (e.g. Hiragana, Katakana, Cyrillic, Greek, Arabic, Hebrew) is selected, that the group be locked in this position until another group select or a de-selection is done (e.g., after Hiragana is selected, returning to Group 1 is typically done by explicitly deselecting Hiragana). The exact way to activate the group selection with a Group Select function is not standardized at this point. It is recommended that at the minimum any Group locking, except for group 1 and group 2, be visually indicated by an appropriate means (e.g. lamp, LCD or screen indication). Ideally the actual group in use should at any time be identified to the user.

- Usual combinations of certain function keys and their standardized interpretation:

  a) *Level 2 Select* + *Level 3 Select* (Shift+AltGr) shall be interpreted as follows (two scenarios are possible) :

  1. According to ISO/IEC 9995-2 : « Specifically, for the harmonized 48 graphic key keyboard arrangement, when characters are allocated in more than one group, the *Group select* function shall be activated by holding a *Level 3 select* key *[AltGr]* depressed while depressing a *Level 2 select [Shift]* key or vice-versa. » One will understand that releasing these two keys without hitting a third key at the same time shall have the same effect as hitting a dedicated *Group Select* key.

  2. If an alphanumeric key is hit while the *Level 2 Select* and *Level 3 Select* are depressed, the state of the keyboard shall be interpreted as providing access to a group related to the previously active group, but different, at level 1. Certain implementations consider this as equivalent to a virtual level 4 in the active group. Although that does not correspond to a standardized concept, this view is tolerable in a restricted linguistic circle.

  b) *Level 2 Select [Shift]* + *Control* shall be interpreted as follows:

© ISO/IEC 2005 – All rights reserved

If an alphanumeric key is hit while the *Level 2 Select [Shift]* and *Control* are depressed, the state of the keyboard shall be interpreted as providing access to a group related to the previously active group, but different, at level 2. Certain implementations consider this as equivalent to a virtual level 5 in the active group. Although that does not correspond to a standardized concept, this view is tolerable in a restricted linguistic circle.

c) *Level 3 Select [AltGr]* + *Control* shall be interpreted as follows:

If an alphanumeric key is hit while the *Level 3 Select* and *Control* are depressed, the state of the keyboard shall be interpreted as providing access to a group related to the previously active group, but different, at level 3. Certain implementations consider this as equivalent to a virtual level 6 in the active group. Although that does not correspond to a standardized concept, this view is tolerable in a restricted linguistic circle.

d) *Level 2 Select [Shift]* + *Group Select* shall be interpreted as follows:

If an alphanumeric key is hit while the *Level 2 Select* and *Group Select* are depressed, the state of the keyboard shall be interpreted as providing access to a group related to the previously active group, but different, at level 2. Certain implementations consider this as equivalent to a virtual level 5 in the active group. Although that does not correspond to a standardized concept, this view is tolerable in a restricted linguistic circle. As an example, in the context of ISO/IEC 9995-3, where group 1 is a Latin national group and where the related group is group 2 (« *Common Secondary Group Layout* » according to ISO/IEC 9995-3 nomenclature), the state of the keyboard locates the next character entry in Group 2, at level 2.

e) *Level 3 Select [AltGr]* + *Group Select* shall be interpreted as follows:

If an alphanumeric key is hit while the *Level 3 Select* and *Group Select* are depressed, the state of the keyboard shall be interpreted as providing access to a group related to the previously active group, but different, at level 3. Certain implementations consider this as equivalent to a virtual level 6 in the active group. Although that does not correspond to a standardized concept, this view is tolerable in a restricted linguistic circle. As an example, in the context of ISO/IEC 9995-3, where group 1 is a Latin national group and where the related group is group 2 (« *Common Secondary Group Layout* » according to ISO/IEC 9995-3 nomenclature), the state of the keyboard locates the next character entry in Group 2, at level 3.

5.3 Special keys

The following function keys exist on different commercialized keyboards. Their type can be generalized according to needs.

a) In Japan, texts can be edited in phonetic script (kanas) or in kanji (Chinese characters). In Korea a similar method is used to convert hangul (Korean letters) into hanza (Chinese characters). A kana-kanji (or hangul-hanza) conversion key allows converting a series of phonetic characters already entered into Chinese characters (kanji or hanza) on the fly. This key is not used to enter characters in conjunction with other keys but must be described as a special function key that involves a transformation program for already entered characters.

b) On certain keyboards outside of Japan or Korea, there exists a combine key that plays a role similar to the kana-kanji key but which is generalized for the generation of characters unavailable on the keyboard from characters already entered on it (fictional example : generating character § from the merging of the two characters s and s, or S and S, or from any combination of these two characters). The functioning of this combine key is not standardized at the moment. On certain keyboards, composition is transitive while on others it is more of a static nature. As for the kana-kanji or hangul-hanza conversion key, hitting the combine key involves a transformation program for already entered characters.

© ISO/IEC 2005 – All rights reserved

5.4 Dead keys

Certain keys are called « dead keys » because hitting them generates a partial character which may or may not appear on the screen. The complete character (also called « fully formed » or « precomposed ») is generally made of a base letter and of one or many diacritical marks.

NOTE International Standard ISO/IEC 9995-3 says what follows about generating fully-formed characters with the help of dead keys:

« Diacritical marks appear above or below certain letters, and all of them are non-spacing characters. Actuating a key with a diacritical mark, followed by actuating a key with a letter, shall indicate that the graphical symbols of the two characters are intended to be combined. Actuating a key with a diacritical mark, followed by actuating the space bar, shall indicate that the diacritical mark is intended to appear as a graphic character of its own (i.e. free-standing).

It is recommended that the method used for the deletion of a character should also be used to cancel a partially-constructed character, such as a diacritical mark without a following letter or a following Space character. »

Note: Diacritical marks may also appear everywhere around the body of a letter or even inside it.

To this can be added that dead key usage may also be transitive, like for the case of combine keys, i.e. certain scripts (polytonic Greek, Vietnamese) require the possibility to enter more than one diacritical mark on a single base letter. Unless the base letters are directly available on the keyboard, precomposed with a first diacritical mark integrated to the letter (as in the case of Scandinavian å, which can also be affected by other diacritical marks), it is then necessary that the function of entering diacritical marks be transitive, i.e. the entry of multiple diacritical marks in a row be applicable to the base character which comes after these marks..

5.5 Stickiness and stable (explicit) locking or temporary (implicit) latching

The state of a keyboard may be locked in a stable or temporary fashion, generally after activating a key or a key sequence..

5.5.1 Stable locking (explicit)

The following keys or functions activate a stable locking, which is deactivated in pressing another time on the same key or in calling the same function:

- Capitals lock
- Level 2 lock
- Group lock
- Numeric lock

5.5.2 Temporary latching (implicit)

International Standard ISO/CEI 9995-3 recommends, when one presses the *Group selection* key, that the state of the keyboard be locked for the entering of the next key only (see also 5.2 under *Group select*). In fact this behaviour depends on the nature of the group invoked, and could apply to groups other than group 2. Group 2 has dead keys whose accents are applicable to basic Latin characters normally found in the group that prevailed before invoking group 2. It is logical to go back to the preceding group for hitting this key. Other characters of group 2 are also infrequently used characters.

Another group which contains infrequent characters may have this temporary latching property. The group definition must then have provision for this implicit temporary latching property (otherwise the group shall be locked explicitly).

© ISO/IEC 2005 – All rights reserved

5.5.3 Stickiness (accessibility)

Stickiness is a temporary function latching property applicable to function keys normally used in conjunction with other keys. The first use is to allow handicapped people to hit keys one by one, without having to type two keys simultaneously (for example for entering the initial capital letter of a sentence, the fact to hit the *Level 2 select* key in this mode will lock the state of the keyboard in capitals just for entering the next key).

In « sticky » mode, successively hitting function keys and releasing them, one by one, is equivalent to virtually maintaining a pressure on all these keys at the same time as long as an alphanumeric key has not been hit. At this moment only, the virtual depression of these function keys is deactivated..

The stickiness mode is activated in depressing a *Level 2 select* key five times in a row. It is deactivated in depressing again a *Level 2 select* key five times in a row.

5.6 Supplementary keys (accessible or not by the computer software) ; Email, Fn function keys, « Windows » keys, Sleep, On/Off)

Even if these keys send a scan code already processed by the computer, they shall be described like other keys, since technically they do not constitute exceptions to what the keyboard description could contain.

Other keys (those not sending a scan code to the computer) may be described narratively as the software might have to be aware of their existence and geometric placement on the keyboard for helping the user.

5.7 Complex state change (mainly for accessibility purposes)

This clause standardizes the following state changes:

Successively hitting *Level 2 select* five times in a row :

Hitting *Level 2 select* during 10 seconds :

[*Alt*][*Level 2 select*][*Print Screen*] :

etc.

5.8 Keyboard feedback

A protocol for the operating systems is specified in Annex A to precisely enquire the hardware for the precise keyboard model used. The response from the keyboard hardware is with a file respecting the keyboard description format specified in clause 5.9. This functionality is optional in this International Standard, and it implements "plug-and-play" functionality for keyboards.

5.9 Machine-readable keyboard description language

The keyboard description format is meant to be capable of describing existing capabilities of today’s keyboard hardware and its associated software, plus foreseeable extensions. It is therefore desirable to define the format in an extensible international standard format like ISO SGML, in the form known as *ISO RELAX NG*, with an easy conversion to industry standard XML. The format is described in Annex B (normative).

The keyboard definition format is primarily intended to be used by the operating system, and during its boot process (eg. in the BIOS), but can also be used for other purposes, such as reporting from the hardware of a keyboard to help the operating system configuring the keyboard driver, or to present the keyboard on the screen with a user-friendly picture.

© ISO/IEC 2005 – All rights reserved

A good test whether the format is capable of supporting existing software is to do a mapping to predominant keyboard description formats and techniques such as Microsoft keyboard definitions, X keyboard definitions, UNIX command line keyboard definitions and industry standard XML. A description of different industry standard formal keyboard description formats and their correspondence to the formal keyboard description format defined in this International Standard is included in Annex D (informative).

In addition some functionality found in some products are covered, such as keyboards with programmable keys and keyboards with multiple key assignments such as telephone keypads.

The keyboard definition format is described in 4 sections:

1. a keyboard identification and general features section, including make and model, serial number, country or region and language to which the keyboard applies, engraving language identification, and distinctive features, such as relief, or presence of lights on keys.

2. the hardware geometry layout, which indicates a largely known geometry layout, such as a 102-key PC keyboard. This section also gives physical information such as size of keys, and amount of pressure needed to activate keys.

3. the keyboard layout, which gives the actual assignment of characters to each key.

4. key combinations which gives combinations of keys, such as those of characters affected by dead keys.

Any of the information may be left out, and the operating system or the user may override the information according to preferences.
The order of precedence in the information modification is first the user, then the system and then the description coming from the hardware.

In the informative Appendix E a number of existing keyboard definition formats is described, together with a mapping between these description formats and the format defined in this International Standard.

© ISO/IEC 2005 – All rights reserved

© ISO/IEC 2005 – All rights reserved

Annex A (normative) – Protocol for the exchange of information between the hardware keyboard and the software

A protocol is defined here to make a conforming keyboard report its configuration to the operating system.

The command to ask the keyboard to report its configuration is issued by the operating system via the sequence:

Set caps Lock
set num lock
clear num lock
set num lock
clear caps lock
clear num lock

Note: Refer to Annex C for a description of
some different stages in history of keyboard hardware and associated software.

© ISO/IEC 2005 – All rights reserved

Annex B (normative) – Formal description language

This Annex specifies semantic and syntax for the keyboard description format defined in this International Standard.

B.1 Description of the formats

To be supplied

B2. A specification in RELAX NG of the keyboard description format

```
default namespace = "http://www.iso.org/WG1"

namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"



# A reference to a set if modifiers

setOfModifiers =

  element modifier {

    attribute name { xsd:string },

    empty

  }+

# Some actions affect either group or

# modifiers

stateChangeAction =

  (# Group management.

   # "relative" attribute

   # indicates whether

   # the value should be

   # treated as absolute or

   # added/removed to the

   # current value

   attribute type { "group" },
```


© ISO/IEC 2005 – All rights reserved

```
    attribute relative { xsd:boolean },
    xsd:integer)
  | (# The list of modifiers to
     # set/latch/lock
     attribute type { "modifiers" },
     setOfModifiers)
# How indicators use groups or modifiers
indicatorUse =
  element use {
    attribute which {
      "base" | "compat" | "effective" | "latched" | "locked"
    },
    empty
  }+
customProperties =
  # Any other stuff - custom properties, if necessary.
  # Can be used in drivers or vendor-specific apps.
  element customProperty {
    attribute name { xsd:string },
    xsd:string
  }*
start =
  element keyboard {
    # First, all the dictionaries we use in the definitions
    element dictionaries {
      # The list of keycodes.
      # The default list of them should be provided separately
      element keycodes {
```

© ISO/IEC 2005 – All rights reserved

```
    # Each keycode is represented by the string id
    # and numeric value
    element keycode {
      (attribute id { xsd:string }
       & attribute value { xsd:int }),
      empty
    }+,
    # The List of aliases
    element alias {
      (attribute id { xsd:string }
       & attribute ref { xsd:string }),
      empty
    }*
  },
  # The list of modifiers used in the definition
  element modifiers {
    # Each modifier has nothing but name
    element modifier {
      attribute name { xsd:string },
      empty
    }*
  },
  # Keyboard types
  element types {
    element type {
      (# Each type has a name
       attribute name { xsd:string }
       & # Number of levels for a type
```


© ISO/IEC 2005 – All rights reserved

```
        attribute numLevels { xsd:int }),
    # Map: (modifiers)->level
    element map {
      element mapEntry {
        attribute level { xsd:int },
        # Set of modifiers (can be empty)
        # Should refer the elements from the modifiers dictionary
        element modifier {
          attribute name { xsd:string },
          empty
        }*
      }*
    }
  }*
},
# Interpretation rules
element interpretation {
  element rule {
    # Each rule has a condition of execution
    element condition {
      # Which keycode this rule is for
      element keysym {
        attribute id { xsd:string },
        empty
      },
      # The modifiers list (for the predicate attribute).
      element modifiersList {
        attribute predicate {
```

© ISO/IEC 2005 – All rights reserved

```
            "all" | "any" | "exactly" | "none"
          }?,
          element modifier {
            attribute name { xsd:string },
            empty
          }+
        }?
      },
      # When the rule condition is met - what the system has to do
      element action {
        # The action is either of ...
        element latch { stateChangeAction }
        | element lock { stateChangeAction }
        | element set { stateChangeAction }
      }
    }*
  }
},
# Meta-information - describes the keyboard purpose
element meta {
  # List of countries, can be empty. ISO 3166
  element countriesList {
    element country { xsd:string }*
  },
  # List of languages, can be empty. ISO 639
  element languagesList {
    element language { xsd:string }*
  },
```


© ISO/IEC 2005 – All rights reserved

```
  customProperties
},
element nonFunctional {
  element geometry { external "Tiny-1.2.rnc" }
},
element functional {
  element groupList {
    element group {
      attribute name { xsd:string }
    }+
  },
  # The list of the keys on the keyboard
  element keyList {
    element key {
      (# Physical scancode
       attribute scancode { xsd:int }
       & # One keycode per key -
         # see the keycodes dictionary above
         attribute keycode { xsd:string }),
      # Meta-information about the key (optional)
      element meta {
        # Engraving (bitmap image)
        element engraving {
          attribute format { "png" | "gif" | "bmp" },
          xsd:base64Binary
        }?,
        customProperties
      }?,
```

© ISO/IEC 2005 – All rights reserved

```
    # Each key can have 0 - N groups
    element group {
      # For each group, a key has a type
      attribute type { xsd:string },
      # Within each group, there can be multiple shift levels
      # On each shift level, we can have either one symbol
      #   or a sequence
      element shiftLevel {
        element symbol { xsd:string }+
      }+
    }*
  }+
},
# The list of indicators
element indicatorsList {
  element indicator {
    # Each real indicator should have a name
    attribute name { xsd:string },
    element groups {
      indicatorUse
      # Which groups are reflected by this indicator

    }?,
    element modifiers {
      indicatorUse,
      # Which modifiers are reflected by this indicator
      setOfModifiers
    }?
```


© ISO/IEC 2005 – All rights reserved

```
        }+
      }
    }
  }
```

B.3 A description in EBNF of the keyboard format

To be supplied

B.4 An example keyboard description

To be supplied

© ISO/IEC 2005 – All rights reserved

Annex C (informative) – Brief History of computer keyboards and their associated software

The technical evolution of PC keyboards have been through a number of phases, in the following order: 1, the original IBM PC, 2, the AT keyboard, and 3: the USB interface. In the following a description of the technical facilities of each of these phases of the interface is presented.

A.1 The IBM keyboard

The IBM keyboard was introduced with the IBM PC in 1981. Of course a number of computer keyboards have existed before that time.

There was a small CPU imbedded in the keyboard, which controlled the keys and the sending of codes to the computer. The codes were the so-called scancodes, a value between 1 and 127. In addition a state code was submitted for every key punched, namely a “push” code when the key was pressed, and a “release” code when the key was released. The CPU was also responsible for generating repeated scancodes, when a key was pressed for a longer time. The state (push or release) was using one bit in front of the one-octet scan code being sent to the computer.

The keyboard cpu could also get an order from the computer to turn on the status of 3 lock controls, namely numeric lock (Num Lock), Caps Lock, and Scroll Lock. This also turned on a light on the keyboard for the corresponding key could be lit or turned off.

The interface between the computer and the keyboard were either a 9 pin DB9 serial interface, or a bigger 5 pin round keyboard plug. Both conformed to the V.24 interface, and there were mechanica converter cables and plugs to convert one plug to another, which only were electrical connections of corresponding pins. The interface speed was typically 9600 bits per second.

The physical layout of the IBM PC keyboard was with a row of 10 function keys at the left side, then a main section with a varying number of keys, depending on the market (country, language) to be covered. The original PC had a single editing and numeric section, and the way to switch between each other was through the NumLock key.. There were no function or special keys in the top of the keyboard.

A.1 The AT and PS/2 keyboard

The IBM PC AT keyboard was much alike the IBM PC keyboard. It also sent keyscan and action codes to the computer, and received state change commands from the computer.

The difference was in the layout of the keyboard, where the 10 function keys to the left were moved to the top of the keyboard, and there were now 2 more (12 function keys). Furthermore an independent editing section was available while the numeric section preserved its compatibility with the first PC (toggling from an editing section to a numeric keybad was still possible through the use of the NumLock key).

The scan codes and the state codes were the transmitted on two octets rather than one octet containing both.

The PS/2 keyboard plug was a smaller round PS/2 plug with 6 pins, which were electrically compatible both with the DB9 interface and the bigger IBM PC round 5-pin plug. The PS/2 plug was often coloured blue, and was often flat or otherwise marked on the top side. Converter cables between the different plug formats existed and were common.

© ISO/IEC 2005 – All rights reserved

A.1 The USB keyboard

The USB keyboard introduced a new generation of electronics for this unit. The physical layout of the keyboard was similar to the AT keyboard, with minimal modifications, but what was sent from the keyboard to the computer was different. The key code values could now be well above 127. Also more information about the keyboard could be obtained via the USB interface. Somehow the interface must be electrically and also codewise compatible, as there exists mechanical plug converters between the USB and the PS/2 interfaces.

© ISO/IEC 2005 – All rights reserved

Annex D (informative) – Format Mapping

This Annex surveys descriptions of machine parsable keyboard description formats, that are used in different operating systems.

For each description format, first an overview of the format is given, then an example description is given, then a formal description in BNF is given and lastly a mapping between the format defined in this international standard and the industry standard description format is given. The information tries to be complete, but there may be errors in the description.

E.1 Microsoft keyboard drivers

To be supplied

The text will include an example generated by the MKLC program, distributed freely by Microsoft at http://www.microsoft.com/globaldev/tools/msklc.mspx

E.2 Unix/Linux X wondow xkb keybord format

The xkb format is part of the X Window graphical system.

E.2.1 xkb descriptions

To be supplied

E.2.2 xkb example

This example is a demonstration example, that is actually not a production specification, but produced for this International Standard. It is believed that the example demonstrates all uses of xkb facilities.

```
xkb_keymap {

xkb_keycodes "xfree86+aliases(qwerty)" {

    minimum = 8;

    maximum = 255;

     <ESC> = 9;

    <AE01> = 10;

    <AE02> = 11;

    <AE03> = 12;

    <AE04> = 13;

    <AE05> = 14;
```


© ISO/IEC 2005 – All rights reserved

<AE06> = 15;

<AE07> = 16;

<AE08> = 17;

<AE09> = 18;

<AE10> = 19;

<AE11> = 20;

<AE12> = 21;

<BKSP> = 22;

<TAB> = 23;

<AD01> = 24;

<AD02> = 25;

<AD03> = 26;

<AD04> = 27;

<AD05> = 28;

<AD06> = 29;

<AD07> = 30;

<AD08> = 31;

<AD09> = 32;

<AD10> = 33;

<AD11> = 34;

<AD12> = 35;

<RTRN> = 36;

<LCTL> = 37;

<AC01> = 38;

<AC02> = 39;

<AC03> = 40;

<AC04> = 41;

<AC05> = 42;

© ISO/IEC 2005 – All rights reserved

```
    <AC06> = 43;
    <AC07> = 44;
    <AC08> = 45;
    <AC09> = 46;
    <AC10> = 47;
    <AC11> = 48;
    <TLDE> = 49;
    <LFSH> = 50;
    <BKSL> = 51;
    <AB01> = 52;
    <AB02> = 53;
    <AB03> = 54;
    <AB04> = 55;
    <AB05> = 56;
    <AB06> = 57;
    <AB07> = 58;
    <AB08> = 59;
    <AB09> = 60;
    <AB10> = 61;
    <RTSH> = 62;
    <KPMU> = 63;
    <LALT> = 64;
    <SPCE> = 65;
    <CAPS> = 66;
    <FK01> = 67;
    <FK02> = 68;
    <FK03> = 69;
    <FK04> = 70;
```


© ISO/IEC 2005 – All rights reserved

<FK05> = 71;

<FK06> = 72;

<FK07> = 73;

<FK08> = 74;

<FK09> = 75;

<FK10> = 76;

<NMLK> = 77;

<SCLK> = 78;

<KP7> = 79;

<KP8> = 80;

<KP9> = 81;

<KPSU> = 82;

<KP4> = 83;

<KP5> = 84;

<KP6> = 85;

<KPAD> = 86;

<KP1> = 87;

<KP2> = 88;

<KP3> = 89;

<KP0> = 90;

<KPDL> = 91;

<SYRQ> = 92;

<MDSW> = 93;

<LSGT> = 94;

<FK11> = 95;

<FK12> = 96;

<HOME> = 97;

<UP> = 98;

© ISO/IEC 2005 – All rights reserved

<PGUP> = 99;

<LEFT> = 100;

<RGHT> = 102;

<END> = 103;

<DOWN> = 104;

<PGDN> = 105;

<INS> = 106;

<DELE> = 107;

<KPEN> = 108;

<RCTL> = 109;

<PAUS> = 110;

<PRSC> = 111;

<KPDV> = 112;

<RALT> = 113;

<BRK> = 114;

<LWIN> = 115;

<RWIN> = 116;

<MENU> = 117;

<FK13> = 118;

<FK14> = 119;

<FK15> = 120;

<FK16> = 121;

<FK17> = 122;

<KPDC> = 123;

<LVL3> = 124;

<ALT> = 125;

<KPEQ> = 126;

<SUPR> = 127;

© ISO/IEC 2005 – All rights reserved

<HYPR> = 128;

<XFER> = 129;

<I02> = 130;

<NFER> = 131;

<I04> = 132;

<AE13> = 133;

<I06> = 134;

<I07> = 135;

<I08> = 136;

<I09> = 137;

<I0A> = 138;

<I0B> = 139;

<I0C> = 140;

<I0D> = 141;

<I0E> = 142;

<I0F> = 143;

<I10> = 144;

<I11> = 145;

<I12> = 146;

<I13> = 147;

<I14> = 148;

<I15> = 149;

<I16> = 150;

<I17> = 151;

<I18> = 152;

<I19> = 153;

<I1A> = 154;

<I1B> = 155;

© ISO/IEC 2005 – All rights reserved

<META> = 156;

<K59> = 157;

<I1E> = 158;

<I1F> = 159;

<I20> = 160;

<I21> = 161;

<I22> = 162;

<I23> = 163;

<I24> = 164;

<I25> = 165;

<I26> = 166;

<I27> = 167;

<I28> = 168;

<I29> = 169;

<K5A> = 170;

<I2B> = 171;

<I2C> = 172;

<I2D> = 173;

<I2E> = 174;

<I2F> = 175;

<I30> = 176;

<I31> = 177;

<I32> = 178;

<I33> = 179;

<I34> = 180;

<K5B> = 181;

<K5D> = 182;

<K5E> = 183;

© ISO/IEC 2005 – All rights reserved

<K5F> = 184;

<I39> = 185;

<I3A> = 186;

<I3B> = 187;

<I3C> = 188;

<K62> = 189;

<K63> = 190;

<K64> = 191;

<K65> = 192;

<K66> = 193;

<I42> = 194;

<I43> = 195;

<I44> = 196;

<I45> = 197;

<K67> = 198;

<K68> = 199;

<K69> = 200;

<K6A> = 201;

<I4A> = 202;

<K6B> = 203;

<K6C> = 204;

<K6D> = 205;

<K6E> = 206;

<K6F> = 207;

<HKTG> = 208;

<K71> = 209;

<K72> = 210;

<AB11> = 211;

© ISO/IEC 2005 – All rights reserved

<I54> = 212;

<I55> = 213;

<I56> = 214;

<I57> = 215;

<I58> = 216;

<I59> = 217;

<I5A> = 218;

<K74> = 219;

<K75> = 220;

<K76> = 221;

<I5E> = 222;

<I5F> = 223;

<I60> = 224;

<I61> = 225;

<I62> = 226;

<I63> = 227;

<I64> = 228;

<I65> = 229;

<I66> = 230;

<I67> = 231;

<I68> = 232;

<I69> = 233;

<I6A> = 234;

<I6B> = 235;

<I6C> = 236;

<I6D> = 237;

<I6E> = 238;

<I6F> = 239;

© ISO/IEC 2005 – All rights reserved

```
    <I70> = 240;
    <I71> = 241;
    <I72> = 242;
    <I73> = 243;
    <I74> = 244;
    <I75> = 245;
    <I76> = 246;
    <I77> = 247;
    <I78> = 248;
    <I79> = 249;
    <I7A> = 250;
    <I7B> = 251;
    <I7C> = 252;
    <I7D> = 253;
    <I7E> = 254;
    <I7F> = 255;
    indicator 1 = "Caps Lock";
    indicator 2 = "Num Lock";
    indicator 3 = "Scroll Lock";
    virtual indicator 4 = "Shift Lock";
    virtual indicator 5 = "Group 2";
    virtual indicator 6 = "Mouse Keys";
    alias <HZTG> = <TLDE>;
    alias <HNGL> = <FK16>;
    alias <HJCV> = <FK17>;
    alias  <I01> = <XFER>;
    alias  <I03> = <NFER>;
    alias  <I05> = <AE13>;
```

© ISO/IEC 2005 – All rights reserved

```
    alias  <K5C> = <KPEQ>;
    alias  <K70> = <HKTG>;
    alias  <K73> = <AB11>;
    alias <LMTA> = <LWIN>;
    alias <RMTA> = <RWIN>;
    alias <COMP> = <MENU>;
    alias <POWR> =  <I0C>;
    alias <MUTE> =  <I0D>;
    alias <VOL-> =  <I0E>;
    alias <VOL+> =  <I0F>;
    alias <HELP> =  <I10>;
    alias <STOP> =  <I11>;
    alias <AGAI> =  <I12>;
    alias <PROP> =  <I13>;
    alias <UNDO> =  <I14>;
    alias <FRNT> =  <I15>;
    alias <COPY> =  <I16>;
    alias <OPEN> =  <I17>;
    alias <PAST> =  <I18>;
    alias <FIND> =  <I19>;
    alias  <CUT> =  <I1A>;
    alias <ALGR> = <RALT>;
    alias <LatQ> = <AD01>;
    alias <LatW> = <AD02>;
    alias <LatE> = <AD03>;
    alias <LatR> = <AD04>;
    alias <LatT> = <AD05>;
    alias <LatY> = <AD06>;
```


© ISO/IEC 2005 – All rights reserved

```
    alias <LatU> = <AD07>;
    alias <LatI> = <AD08>;
    alias <LatO> = <AD09>;
    alias <LatP> = <AD10>;
    alias <LatA> = <AC01>;
    alias <LatS> = <AC02>;
    alias <LatD> = <AC03>;
    alias <LatF> = <AC04>;
    alias <LatG> = <AC05>;
    alias <LatH> = <AC06>;
    alias <LatJ> = <AC07>;
    alias <LatK> = <AC08>;
    alias <LatL> = <AC09>;
    alias <LatZ> = <AB01>;
    alias <LatX> = <AB02>;
    alias <LatC> = <AB03>;
    alias <LatV> = <AB04>;
    alias <LatB> = <AB05>;
    alias <LatN> = <AB06>;
    alias <LatM> = <AB07>;
};

xkb_types "complete" {

    virtual_modifiers NumLock,Alt,LevelThree,ScrollLock,LevelFive,AltGr,Meta,Super,Hyper;

    type "ONE_LEVEL" {
        modifiers= none;
```

© ISO/IEC 2005 – All rights reserved

```
        level_name[Level1]= "Any";
    };
    type "TWO_LEVEL" {
        modifiers= Shift;
        map[Shift]= Level2;
        level_name[Level1]= "Base";
        level_name[Level2]= "Shift";
    };
    type "ALPHABETIC" {
        modifiers= Shift+Lock;
        map[Shift]= Level2;
        map[Lock]= Level2;
        level_name[Level1]= "Base";
        level_name[Level2]= "Caps";
    };
    type "KEYPAD" {
        modifiers= Shift+NumLock;
        map[Shift]= Level2;
        map[NumLock]= Level2;
        level_name[Level1]= "Base";
        level_name[Level2]= "Number";
    };
    type "SHIFT+ALT" {
        modifiers= Shift+Alt;
        map[Shift+Alt]= Level2;
        level_name[Level1]= "Base";
        level_name[Level2]= "Shift+Alt";
    };
```


© ISO/IEC 2005 – All rights reserved

```
type "PC_BREAK" {
    modifiers= Control;
    map[Control]= Level2;
    level_name[Level1]= "Base";
    level_name[Level2]= "Control";
};
type "PC_SYSRQ" {
    modifiers= Alt+LevelThree;
    map[Alt]= Level2;
    map[LevelThree]= Level3;
    level_name[Level1]= "Base";
    level_name[Level2]= "Alt";
    level_name[Level3]= "Level3";
};
type "CTRL+ALT" {
    modifiers= Control+Alt;
    map[Control+Alt]= Level2;
    level_name[Level1]= "Base";
    level_name[Level2]= "Ctrl+Alt";
};
type "THREE_LEVEL" {
    modifiers= Shift+LevelThree;
    map[Shift]= Level2;
    map[LevelThree]= Level3;
    map[Shift+LevelThree]= Level3;
    level_name[Level1]= "Base";
    level_name[Level2]= "Shift";
    level_name[Level3]= "Level3";
```

© ISO/IEC 2005 – All rights reserved

```
};
type "EIGHT_LEVEL" {
    modifiers= Shift+LevelThree+LevelFive;
    map[Shift]= Level2;
    map[LevelThree]= Level3;
    map[Shift+LevelThree]= Level4;
    map[LevelFive]= Level5;
    map[Shift+LevelFive]= Level6;
    map[LevelThree+LevelFive]= Level7;
    map[Shift+LevelThree+LevelFive]= Level8;
    level_name[Level1]= "Base";
    level_name[Level2]= "Shift";
    level_name[Level3]= "Alt Base";
    level_name[Level4]= "Shift Alt";
    level_name[Level5]= "X";
    level_name[Level6]= "X Shift";
    level_name[Level7]= "X Alt Base";
    level_name[Level8]= "X Shift Alt";
};
type "EIGHT_LEVEL_ALPHABETIC" {
    modifiers= Shift+Lock+LevelThree+LevelFive;
    map[Shift]= Level2;
    map[Lock]= Level2;
    map[LevelThree]= Level3;
    map[Shift+LevelThree]= Level4;
    map[Lock+LevelThree]= Level4;
    map[Shift+Lock+LevelThree]= Level3;
    map[LevelFive]= Level5;
```


© ISO/IEC 2005 – All rights reserved

```
        map[Shift+LevelFive]= Level6;
        map[Lock+LevelFive]= Level6;
        map[LevelThree+LevelFive]= Level7;
        map[Shift+LevelThree+LevelFive]= Level8;
        map[Lock+LevelThree+LevelFive]= Level8;
        map[Shift+Lock+LevelThree+LevelFive]= Level7;
        level_name[Level1]= "Base";
        level_name[Level2]= "Shift";
        level_name[Level3]= "Alt Base";
        level_name[Level4]= "Shift Alt";
        level_name[Level5]= "X";
        level_name[Level6]= "X Shift";
        level_name[Level7]= "X Alt Base";
        level_name[Level8]= "X Shift Alt";
    };
    type "EIGHT_LEVEL_SEMIALPHABETIC" {
        modifiers= Shift+Lock+LevelThree+LevelFive;
        map[Shift]= Level2;
        map[Lock]= Level2;
        map[LevelThree]= Level3;
        map[Shift+LevelThree]= Level4;
        map[Lock+LevelThree]= Level3;
        preserve[Lock+LevelThree]= Lock;
        map[Shift+Lock+LevelThree]= Level4;
        preserve[Shift+Lock+LevelThree]= Lock;
        map[LevelFive]= Level5;
        map[Shift+LevelFive]= Level6;
        map[Lock+LevelFive]= Level6;
```

© ISO/IEC 2005 – All rights reserved

```
        preserve[Lock+LevelFive]= Lock;
        map[LevelThree+LevelFive]= Level7;
        map[Shift+LevelThree+LevelFive]= Level8;
        map[Lock+LevelThree+LevelFive]= Level7;
        preserve[Lock+LevelThree+LevelFive]= Lock;
        map[Shift+Lock+LevelThree+LevelFive]= Level8;
        preserve[Shift+Lock+LevelThree+LevelFive]= Lock;
        map[Shift+Lock+LevelFive]= Level1;
        preserve[Shift+Lock+LevelFive]= Lock;
        level_name[Level1]= "Base";
        level_name[Level2]= "Shift";
        level_name[Level3]= "Alt Base";
        level_name[Level4]= "Shift Alt";
        level_name[Level5]= "X";
        level_name[Level6]= "X Shift";
        level_name[Level7]= "X Alt Base";
        level_name[Level8]= "X Shift Alt";
    };
    type "FOUR_LEVEL" {
        modifiers= Shift+LevelThree;
        map[Shift]= Level2;
        map[LevelThree]= Level3;
        map[Shift+LevelThree]= Level4;
        level_name[Level1]= "Base";
        level_name[Level2]= "Shift";
        level_name[Level3]= "Alt Base";
        level_name[Level4]= "Shift Alt";
    };
```


© ISO/IEC 2005 – All rights reserved

```
type "FOUR_LEVEL_ALPHABETIC" {
    modifiers= Shift+Lock+LevelThree;
    map[Shift]= Level2;
    map[Lock]= Level2;
    map[LevelThree]= Level3;
    map[Shift+LevelThree]= Level4;
    map[Lock+LevelThree]= Level4;
    map[Shift+Lock+LevelThree]= Level3;
    level_name[Level1]= "Base";
    level_name[Level2]= "Shift";
    level_name[Level3]= "Alt Base";
    level_name[Level4]= "Shift Alt";
};
type "FOUR_LEVEL_SEMIALPHABETIC" {
    modifiers= Shift+Lock+LevelThree;
    map[Shift]= Level2;
    map[Lock]= Level2;
    map[LevelThree]= Level3;
    map[Shift+LevelThree]= Level4;
    map[Lock+LevelThree]= Level3;
    preserve[Lock+LevelThree]= Lock;
    map[Shift+Lock+LevelThree]= Level4;
    preserve[Shift+Lock+LevelThree]= Lock;
    level_name[Level1]= "Base";
    level_name[Level2]= "Shift";
    level_name[Level3]= "Alt Base";
    level_name[Level4]= "Shift Alt";
};
```

© ISO/IEC 2005 – All rights reserved

```
    type "FOUR_LEVEL_KEYPAD" {
        modifiers= Shift+NumLock+LevelThree;
        map[Shift]= Level2;
        map[NumLock]= Level2;
        map[LevelThree]= Level3;
        map[Shift+LevelThree]= Level4;
        map[NumLock+LevelThree]= Level4;
        map[Shift+NumLock+LevelThree]= Level3;
        level_name[Level1]= "Base";
        level_name[Level2]= "Number";
        level_name[Level3]= "Alt Base";
        level_name[Level4]= "Alt Number";
    };
    type "SEPARATE_CAPS_AND_SHIFT_ALPHABETIC" {
        modifiers= Shift+Lock+LevelThree;
        map[Shift]= Level2;
        map[Lock]= Level4;
        preserve[Lock]= Lock;
        map[LevelThree]= Level3;
        map[Shift+LevelThree]= Level4;
        map[Lock+LevelThree]= Level3;
        preserve[Lock+LevelThree]= Lock;
        map[Shift+Lock+LevelThree]= Level3;
        level_name[Level1]= "Base";
        level_name[Level2]= "Shift";
        level_name[Level3]= "AltGr Base";
        level_name[Level4]= "Shift AltGr";
    };
```


© ISO/IEC 2005 – All rights reserved

```
};

xkb_compatibility "complete" {

    virtual_modifiers NumLock,Alt,LevelThree,ScrollLock,LevelFive,AltGr,Meta,Super,Hyper;

    interpret.useModMapMods= AnyLevel;
    interpret.repeat= False;
    interpret.locking= False;
    interpret ISO_Level2_Latch+Exactly(Shift) {
        useModMapMods=level1;
        action= LatchMods(modifiers=Shift,clearLocks,latchToLock);
    };
    interpret Shift_Lock+AnyOf(Shift+Lock) {
        action= LockMods(modifiers=Shift);
    };
    interpret Num_Lock+AnyOf(all) {
        virtualModifier= NumLock;
        action= LockMods(modifiers=NumLock);
    };
    interpret ISO_Lock+AnyOf(all) {
        action= ISOLock(modifiers=modMapMods,affect=all);
    };
    interpret ISO_Level3_Shift+AnyOf(all) {
        virtualModifier= LevelThree;
        useModMapMods=level1;
        action= SetMods(modifiers=LevelThree,clearLocks);
    };
```

© ISO/IEC 2005 – All rights reserved

```
    interpret ISO_Level3_Latch+AnyOf(all) {
        virtualModifier= LevelThree;
        useModMapMods=level1;
        action= LatchMods(modifiers=LevelThree,clearLocks,latchToLock);
    };
    interpret ISO_Level3_Lock+AnyOf(all) {
        virtualModifier= LevelThree;
        useModMapMods=level1;
        action= LockMods(modifiers=LevelThree);
    };
    interpret Alt_L+AnyOf(all) {
        virtualModifier= Alt;
        action= SetMods(modifiers=modMapMods,clearLocks);
    };
    interpret Alt_R+AnyOf(all) {
        virtualModifier= Alt;
        action= SetMods(modifiers=modMapMods,clearLocks);
    };
    interpret Meta_L+AnyOf(all) {
        virtualModifier= Meta;
        action= SetMods(modifiers=modMapMods,clearLocks);
    };
    interpret Meta_R+AnyOf(all) {
        virtualModifier= Meta;
        action= SetMods(modifiers=modMapMods,clearLocks);
    };
    interpret Super_L+AnyOf(all) {
        virtualModifier= Super;
```


© ISO/IEC 2005 – All rights reserved

```
    action= SetMods(modifiers=modMapMods,clearLocks);
};
interpret Super_R+AnyOf(all) {
    virtualModifier= Super;
    action= SetMods(modifiers=modMapMods,clearLocks);
};
interpret Hyper_L+AnyOf(all) {
    virtualModifier= Hyper;
    action= SetMods(modifiers=modMapMods,clearLocks);
};
interpret Hyper_R+AnyOf(all) {
    virtualModifier= Hyper;
    action= SetMods(modifiers=modMapMods,clearLocks);
};
interpret Scroll_Lock+AnyOf(all) {
    virtualModifier= ScrollLock;
    action= LockMods(modifiers=modMapMods);
};
interpret F35+AnyOf(all) {
    virtualModifier= LevelFive;
    useModMapMods=level1;
    action= SetMods(modifiers=LevelFive,clearLocks);
};
interpret F34+AnyOf(all) {
    virtualModifier= LevelFive;
    action= LatchMods(modifiers=LevelFive,clearLocks,latchToLock);
};
interpret F33+AnyOf(all) {
```

© ISO/IEC 2005 – All rights reserved

```
        virtualModifier= LevelFive;
        action= LockMods(modifiers=LevelFive);
    };
    interpret Mode_switch+AnyOfOrNone(all) {
        virtualModifier= AltGr;
        useModMapMods=level1;
        action= SetGroup(group=+1);
    };
    interpret ISO_Level3_Shift+AnyOfOrNone(all) {
        action= SetMods(modifiers=LevelThree,clearLocks);
    };
    interpret ISO_Level3_Latch+AnyOfOrNone(all) {
        action= LatchMods(modifiers=LevelThree,clearLocks,latchToLock);
    };
    interpret ISO_Level3_Lock+AnyOfOrNone(all) {
        action= LockMods(modifiers=LevelThree);
    };
    interpret ISO_Group_Latch+AnyOfOrNone(all) {
        virtualModifier= AltGr;
        useModMapMods=level1;
        action= LatchGroup(group=2);
    };
    interpret ISO_Next_Group+AnyOfOrNone(all) {
        virtualModifier= AltGr;
        useModMapMods=level1;
        action= LockGroup(group=+1);
    };
    interpret ISO_Prev_Group+AnyOfOrNone(all) {
```


© ISO/IEC 2005 – All rights reserved

```
    virtualModifier= AltGr;
    useModMapMods=level1;
    action= LockGroup(group=-1);
};
interpret ISO_First_Group+AnyOfOrNone(all) {
    action= LockGroup(group=1);
};
interpret ISO_Last_Group+AnyOfOrNone(all) {
    action= LockGroup(group=2);
};
interpret KP_1+AnyOfOrNone(all) {
    repeat= True;
    action= MovePtr(x=-1,y=+1);
};
interpret KP_End+AnyOfOrNone(all) {
    repeat= True;
    action= MovePtr(x=-1,y=+1);
};
interpret KP_2+AnyOfOrNone(all) {
    repeat= True;
    action= MovePtr(x=+0,y=+1);
};
interpret KP_Down+AnyOfOrNone(all) {
    repeat= True;
    action= MovePtr(x=+0,y=+1);
};
interpret KP_3+AnyOfOrNone(all) {
    repeat= True;
```

© ISO/IEC 2005 – All rights reserved

```
        action= MovePtr(x=+1,y=+1);
    };
    interpret KP_Next+AnyOfOrNone(all) {
        repeat= True;
        action= MovePtr(x=+1,y=+1);
    };
    interpret KP_4+AnyOfOrNone(all) {
        repeat= True;
        action= MovePtr(x=-1,y=+0);
    };
    interpret KP_Left+AnyOfOrNone(all) {
        repeat= True;
        action= MovePtr(x=-1,y=+0);
    };
    interpret KP_6+AnyOfOrNone(all) {
        repeat= True;
        action= MovePtr(x=+1,y=+0);
    };
    interpret KP_Right+AnyOfOrNone(all) {
        repeat= True;
        action= MovePtr(x=+1,y=+0);
    };
    interpret KP_7+AnyOfOrNone(all) {
        repeat= True;
        action= MovePtr(x=-1,y=-1);
    };
    interpret KP_Home+AnyOfOrNone(all) {
        repeat= True;
```


© ISO/IEC 2005 – All rights reserved

```
    action= MovePtr(x=-1,y=-1);
};
interpret KP_8+AnyOfOrNone(all) {
    repeat= True;
    action= MovePtr(x=+0,y=-1);
};
interpret KP_Up+AnyOfOrNone(all) {
    repeat= True;
    action= MovePtr(x=+0,y=-1);
};
interpret KP_9+AnyOfOrNone(all) {
    repeat= True;
    action= MovePtr(x=+1,y=-1);
};
interpret KP_Prior+AnyOfOrNone(all) {
    repeat= True;
    action= MovePtr(x=+1,y=-1);
};
interpret KP_5+AnyOfOrNone(all) {
    repeat= True;
    action= PtrBtn(button=default);
};
interpret KP_Begin+AnyOfOrNone(all) {
    repeat= True;
    action= PtrBtn(button=default);
};
interpret KP_F2+AnyOfOrNone(all) {
    repeat= True;
```

© ISO/IEC 2005 – All rights reserved

```
        action= SetPtrDflt(affect=button,button=1);
    };
    interpret KP_Divide+AnyOfOrNone(all) {
        repeat= True;
        action= SetPtrDflt(affect=button,button=1);
    };
    interpret KP_F3+AnyOfOrNone(all) {
        repeat= True;
        action= SetPtrDflt(affect=button,button=2);
    };
    interpret KP_Multiply+AnyOfOrNone(all) {
        repeat= True;
        action= SetPtrDflt(affect=button,button=2);
    };
    interpret KP_F4+AnyOfOrNone(all) {
        repeat= True;
        action= SetPtrDflt(affect=button,button=3);
    };
    interpret KP_Subtract+AnyOfOrNone(all) {
        repeat= True;
        action= SetPtrDflt(affect=button,button=3);
    };
    interpret KP_Separator+AnyOfOrNone(all) {
        repeat= True;
        action= PtrBtn(button=default,count=2);
    };
    interpret KP_Add+AnyOfOrNone(all) {
        repeat= True;
```


© ISO/IEC 2005 – All rights reserved

```
    action= PtrBtn(button=default,count=2);
};
interpret KP_0+AnyOfOrNone(all) {
    repeat= True;
    action= LockPtrBtn(button=default,affect=lock);
};
interpret KP_Insert+AnyOfOrNone(all) {
    repeat= True;
    action= LockPtrBtn(button=default,affect=lock);
};
interpret KP_Decimal+AnyOfOrNone(all) {
    repeat= True;
    action= LockPtrBtn(button=default,affect=unlock);
};
interpret KP_Delete+AnyOfOrNone(all) {
    repeat= True;
    action= LockPtrBtn(button=default,affect=unlock);
};
interpret F25+AnyOfOrNone(all) {
    repeat= True;
    action= SetPtrDflt(affect=button,button=1);
};
interpret F26+AnyOfOrNone(all) {
    repeat= True;
    action= SetPtrDflt(affect=button,button=2);
};
interpret F27+AnyOfOrNone(all) {
    repeat= True;
```

© ISO/IEC 2005 – All rights reserved

```
        action= MovePtr(x=-1,y=-1);
    };
    interpret F29+AnyOfOrNone(all) {
        repeat= True;
        action= MovePtr(x=+1,y=-1);
    };
    interpret F31+AnyOfOrNone(all) {
        repeat= True;
        action= PtrBtn(button=default);
    };
    interpret F33+AnyOfOrNone(all) {
        repeat= True;
        action= MovePtr(x=-1,y=+1);
    };
    interpret F35+AnyOfOrNone(all) {
        repeat= True;
        action= MovePtr(x=+1,y=+1);
    };
    interpret Pointer_Button_Dflt+AnyOfOrNone(all) {
        action= PtrBtn(button=default);
    };
    interpret Pointer_Button1+AnyOfOrNone(all) {
        action= PtrBtn(button=1);
    };
    interpret Pointer_Button2+AnyOfOrNone(all) {
        action= PtrBtn(button=2);
    };
    interpret Pointer_Button3+AnyOfOrNone(all) {
```


© ISO/IEC 2005 – All rights reserved

```
    action= PtrBtn(button=3);
};
interpret Pointer_DblClick_Dflt+AnyOfOrNone(all) {
    action= PtrBtn(button=default,count=2);
};
interpret Pointer_DblClick1+AnyOfOrNone(all) {
    action= PtrBtn(button=1,count=2);
};
interpret Pointer_DblClick2+AnyOfOrNone(all) {
    action= PtrBtn(button=2,count=2);
};
interpret Pointer_DblClick3+AnyOfOrNone(all) {
    action= PtrBtn(button=3,count=2);
};
interpret Pointer_Drag_Dflt+AnyOfOrNone(all) {
    action= LockPtrBtn(button=default,affect=both);
};
interpret Pointer_Drag1+AnyOfOrNone(all) {
    action= LockPtrBtn(button=1,affect=both);
};
interpret Pointer_Drag2+AnyOfOrNone(all) {
    action= LockPtrBtn(button=2,affect=both);
};
interpret Pointer_Drag3+AnyOfOrNone(all) {
    action= LockPtrBtn(button=3,affect=both);
};
interpret Pointer_EnableKeys+AnyOfOrNone(all) {
    action= LockControls(controls=MouseKeys);
```

© ISO/IEC 2005 – All rights reserved

```
};
interpret Pointer_Accelerate+AnyOfOrNone(all) {
    action= LockControls(controls=MouseKeysAccel);
};
interpret Pointer_DfltBtnNext+AnyOfOrNone(all) {
    action= SetPtrDflt(affect=button,button=+1);
};
interpret Pointer_DfltBtnPrev+AnyOfOrNone(all) {
    action= SetPtrDflt(affect=button,button=-1);
};
interpret AccessX_Enable+AnyOfOrNone(all) {
    action= LockControls(controls=AccessXKeys);
};
interpret AccessX_Feedback_Enable+AnyOfOrNone(all) {
    action= LockControls(controls=AccessXFeedback);
};
interpret RepeatKeys_Enable+AnyOfOrNone(all) {
    action= LockControls(controls=RepeatKeys);
};
interpret SlowKeys_Enable+AnyOfOrNone(all) {
    action= LockControls(controls=SlowKeys);
};
interpret BounceKeys_Enable+AnyOfOrNone(all) {
    action= LockControls(controls=BounceKeys);
};
interpret StickyKeys_Enable+AnyOfOrNone(all) {
    action= LockControls(controls=StickyKeys);
};
```


© ISO/IEC 2005 – All rights reserved

```
interpret MouseKeys_Enable+AnyOfOrNone(all) {
    action= LockControls(controls=MouseKeys);
};
interpret MouseKeys_Accel_Enable+AnyOfOrNone(all) {
    action= LockControls(controls=MouseKeysAccel);
};
interpret Overlay1_Enable+AnyOfOrNone(all) {
    action= LockControls(controls=Overlay1);
};
interpret Overlay2_Enable+AnyOfOrNone(all) {
    action= LockControls(controls=Overlay2);
};
interpret AudibleBell_Enable+AnyOfOrNone(all) {
    action= LockControls(controls=AudibleBell);
};
interpret Terminate_Server+AnyOfOrNone(all) {
    action= Terminate();
};
interpret Alt_L+AnyOfOrNone(all) {
    action= SetMods(modifiers=Alt,clearLocks);
};
interpret Alt_R+AnyOfOrNone(all) {
    action= SetMods(modifiers=Alt,clearLocks);
};
interpret Meta_L+AnyOfOrNone(all) {
    action= SetMods(modifiers=Meta,clearLocks);
};
interpret Meta_R+AnyOfOrNone(all) {
```

© ISO/IEC 2005 – All rights reserved

```
        action= SetMods(modifiers=Alt,clearLocks);
    };
    interpret Super_L+AnyOfOrNone(all) {
        action= SetMods(modifiers=Super,clearLocks);
    };
    interpret Super_R+AnyOfOrNone(all) {
        action= SetMods(modifiers=Super,clearLocks);
    };
    interpret Hyper_L+AnyOfOrNone(all) {
        action= SetMods(modifiers=Hyper,clearLocks);
    };
    interpret Hyper_R+AnyOfOrNone(all) {
        action= SetMods(modifiers=Hyper,clearLocks);
    };
    interpret XF86_Switch_VT_1+AnyOfOrNone(all) {
        repeat= True;
        action= SwitchScreen(screen=1,!same);
    };
    interpret XF86_Switch_VT_2+AnyOfOrNone(all) {
        repeat= True;
        action= SwitchScreen(screen=2,!same);
    };
    interpret XF86_Switch_VT_3+AnyOfOrNone(all) {
        repeat= True;
        action= SwitchScreen(screen=3,!same);
    };
    interpret XF86_Switch_VT_4+AnyOfOrNone(all) {
        repeat= True;
```


© ISO/IEC 2005 – All rights reserved

```
    action= SwitchScreen(screen=4,!same);
};
interpret XF86_Switch_VT_5+AnyOfOrNone(all) {
    repeat= True;
    action= SwitchScreen(screen=5,!same);
};
interpret XF86_Switch_VT_6+AnyOfOrNone(all) {
    repeat= True;
    action= SwitchScreen(screen=6,!same);
};
interpret XF86_Switch_VT_7+AnyOfOrNone(all) {
    repeat= True;
    action= SwitchScreen(screen=7,!same);
};
interpret XF86_Switch_VT_8+AnyOfOrNone(all) {
    repeat= True;
    action= SwitchScreen(screen=8,!same);
};
interpret XF86_Switch_VT_9+AnyOfOrNone(all) {
    repeat= True;
    action= SwitchScreen(screen=9,!same);
};
interpret XF86_Switch_VT_10+AnyOfOrNone(all) {
    repeat= True;
    action= SwitchScreen(screen=10,!same);
};
interpret XF86_Switch_VT_11+AnyOfOrNone(all) {
    repeat= True;
```

© ISO/IEC 2005 – All rights reserved

```
        action= SwitchScreen(screen=11,!same);

    };

    interpret XF86_Switch_VT_12+AnyOfOrNone(all) {

        repeat= True;

        action= SwitchScreen(screen=12,!same);

    };

    interpret XF86_Ungrab+AnyOfOrNone(all) {

        repeat= True;

        action=
Private(type=0x86,data[0]=0x55,data[1]=0x6e,data[2]=0x67,data[3]=0x72,data[4]=0x61,data[5]=0x62,d
ata[6]=0x00);

    };

    interpret XF86_ClearGrab+AnyOfOrNone(all) {

        repeat= True;

        action=
Private(type=0x86,data[0]=0x43,data[1]=0x6c,data[2]=0x73,data[3]=0x47,data[4]=0x72,data[5]=0x62,d
ata[6]=0x00);

    };

    interpret XF86_Next_VMode+AnyOfOrNone(all) {

        repeat= True;

        action=
Private(type=0x86,data[0]=0x2b,data[1]=0x56,data[2]=0x4d,data[3]=0x6f,data[4]=0x64,data[5]=0x65,d
ata[6]=0x00);

    };

    interpret XF86_Prev_VMode+AnyOfOrNone(all) {

        repeat= True;

        action=
Private(type=0x86,data[0]=0x2d,data[1]=0x56,data[2]=0x4d,data[3]=0x6f,data[4]=0x64,data[5]=0x65,d
ata[6]=0x00);

    };

    interpret F34+AnyOfOrNone(all) {

        action= LatchMods(modifiers=LevelFive,clearLocks,latchToLock);
```


© ISO/IEC 2005 – All rights reserved

```
};
interpret Any+Exactly(Lock) {
    action= LockMods(modifiers=Lock);
};
interpret Any+AnyOf(all) {
    action= SetMods(modifiers=modMapMods,clearLocks);
};
group 2 = AltGr;
group 3 = AltGr;
group 4 = AltGr;
indicator "Caps Lock" {
    !allowExplicit;
    whichModState= locked;
    modifiers= Lock;
};
indicator "Num Lock" {
    !allowExplicit;
    whichModState= locked;
    modifiers= NumLock;
};
indicator "Scroll Lock" {
    whichModState= locked;
    modifiers= ScrollLock;
};
indicator "Shift Lock" {
    !allowExplicit;
    whichModState= locked;
    modifiers= Shift;
```

© ISO/IEC 2005 – All rights reserved

```
    };
    indicator "Group 2" {
        !allowExplicit;
        groups= 0xfe;
    };
    indicator "Mouse Keys" {
        indicatorDrivesKeyboard;
        controls= mouseKeys;
    };
};

xkb_symbols "pc(pc105)+us+inet(power_g5)+ru(winkeys):2+group(rctrl_toggle)+eurosign(e)" {

    name[group1]="U.S. English";
    name[group2]="Russia - Winkeys";

    key  <ESC> {         [          Escape ] };
    key <AE01> {         [               1,          exclam ] };
    key <AE02> {
        symbols[Group1]= [               2,              at ],
        symbols[Group2]= [               2,        quotedbl ]
    };
    key <AE03> {
        symbols[Group1]= [               3,      numbersign ],
        symbols[Group2]= [               3,      numerosign ]
    };
    key <AE04> {
        symbols[Group1]= [               4,          dollar ],
```


© ISO/IEC 2005 – All rights reserved

```
    symbols[Group2]= [               4,       semicolon ]
};
key <AE05> {         [               5,         percent ] };
key <AE06> {
    symbols[Group1]= [               6,     asciicircum ],
    symbols[Group2]= [               6,           colon ]
};
key <AE07> {
    symbols[Group1]= [               7,       ampersand ],
    symbols[Group2]= [               7,        question ]
};
key <AE08> {         [               8,        asterisk ] };
key <AE09> {         [               9,       parenleft ] };
key <AE10> {         [               0,      parenright ] };
key <AE11> {         [           minus,      underscore ] };
key <AE12> {         [           equal,            plus ] };
key <BKSP> {
    type= "CTRL+ALT",
    symbols[Group1]= [       BackSpace, Terminate_Server ]
};
key  <TAB> {         [             Tab,    ISO_Left_Tab ] };
key <AD01> {
    type= "ALPHABETIC",
    symbols[Group1]= [               q,               Q ],
    symbols[Group2]= [ Cyrillic_shorti, Cyrillic_SHORTI ]
};
key <AD02> {
    type= "ALPHABETIC",
```

© ISO/IEC 2005 – All rights reserved

```
        symbols[Group1]= [               w,               W ],
        symbols[Group2]= [    Cyrillic_tse,    Cyrillic_TSE ]
    };
    key <AD03> {
        type[group1]= "FOUR_LEVEL_SEMIALPHABETIC",
        type[group2]= "ALPHABETIC",
        symbols[Group1]= [               e,               E,        EuroSign,        NoSymbol ],
        symbols[Group2]= [      Cyrillic_u,      Cyrillic_U ]
    };
    key <AD04> {
        type= "ALPHABETIC",
        symbols[Group1]= [               r,               R ],
        symbols[Group2]= [     Cyrillic_ka,     Cyrillic_KA ]
    };
    key <AD05> {
        type= "ALPHABETIC",
        symbols[Group1]= [               t,               T ],
        symbols[Group2]= [     Cyrillic_ie,     Cyrillic_IE ]
    };
    key <AD06> {
        type= "ALPHABETIC",
        symbols[Group1]= [               y,               Y ],
        symbols[Group2]= [     Cyrillic_en,     Cyrillic_EN ]
    };
    key <AD07> {
        type= "ALPHABETIC",
        symbols[Group1]= [               u,               U ],
        symbols[Group2]= [    Cyrillic_ghe,    Cyrillic_GHE ]
```


© ISO/IEC 2005 – All rights reserved

```
};
key <AD08> {
    type= "ALPHABETIC",
    symbols[Group1]= [               i,               I ],
    symbols[Group2]= [    Cyrillic_sha,    Cyrillic_SHA ]
};
key <AD09> {
    type= "ALPHABETIC",
    symbols[Group1]= [               o,               O ],
    symbols[Group2]= [  Cyrillic_shcha,  Cyrillic_SHCHA ]
};
key <AD10> {
    type= "ALPHABETIC",
    symbols[Group1]= [               p,               P ],
    symbols[Group2]= [     Cyrillic_ze,     Cyrillic_ZE ]
};
key <AD11> {
    type[group2]= "ALPHABETIC",
    symbols[Group1]= [     bracketleft,       braceleft ],
    symbols[Group2]= [     Cyrillic_ha,     Cyrillic_HA ]
};
key <AD12> {
    type[group2]= "ALPHABETIC",
    symbols[Group1]= [    bracketright,      braceright ],
    symbols[Group2]= [ Cyrillic_hardsign, Cyrillic_HARDSIGN ]
};
key <RTRN> {         [          Return ] };
key <LCTL> {         [       Control_L ] };
```

© ISO/IEC 2005 – All rights reserved

```
    key <AC01> {
        type= "ALPHABETIC",
        symbols[Group1]= [               a,               A ],
        symbols[Group2]= [     Cyrillic_ef,     Cyrillic_EF ]
    };
    key <AC02> {
        type= "ALPHABETIC",
        symbols[Group1]= [               s,               S ],
        symbols[Group2]= [   Cyrillic_yeru,   Cyrillic_YERU ]
    };
    key <AC03> {
        type= "ALPHABETIC",
        symbols[Group1]= [               d,               D ],
        symbols[Group2]= [     Cyrillic_ve,     Cyrillic_VE ]
    };
    key <AC04> {
        type= "ALPHABETIC",
        symbols[Group1]= [               f,               F ],
        symbols[Group2]= [      Cyrillic_a,      Cyrillic_A ]
    };
    key <AC05> {
        type= "ALPHABETIC",
        symbols[Group1]= [               g,               G ],
        symbols[Group2]= [     Cyrillic_pe,     Cyrillic_PE ]
    };
    key <AC06> {
        type= "ALPHABETIC",
        symbols[Group1]= [               h,               H ],
```


© ISO/IEC 2005 – All rights reserved

```
    symbols[Group2]= [     Cyrillic_er,     Cyrillic_ER ]
};
key <AC07> {
    type= "ALPHABETIC",
    symbols[Group1]= [               j,               J ],
    symbols[Group2]= [      Cyrillic_o,      Cyrillic_O ]
};
key <AC08> {
    type= "ALPHABETIC",
    symbols[Group1]= [               k,               K ],
    symbols[Group2]= [     Cyrillic_el,     Cyrillic_EL ]
};
key <AC09> {
    type= "ALPHABETIC",
    symbols[Group1]= [               l,               L ],
    symbols[Group2]= [     Cyrillic_de,     Cyrillic_DE ]
};
key <AC10> {
    type[group2]= "ALPHABETIC",
    symbols[Group1]= [       semicolon,           colon ],
    symbols[Group2]= [    Cyrillic_zhe,    Cyrillic_ZHE ]
};
key <AC11> {
    type[group2]= "ALPHABETIC",
    symbols[Group1]= [      apostrophe,        quotedbl ],
    symbols[Group2]= [      Cyrillic_e,      Cyrillic_E ]
};
key <TLDE> {
```

© ISO/IEC 2005 – All rights reserved

```
        type[group2]= "ALPHABETIC",
        symbols[Group1]= [           grave,      asciitilde ],
        symbols[Group2]= [     Cyrillic_io,     Cyrillic_IO ]
    };
    key <LFSH> {         [         Shift_L ] };
    key <BKSL> {
        symbols[Group1]= [       backslash,             bar ],
        symbols[Group2]= [       backslash,           slash ]
    };
    key <AB01> {
        type= "ALPHABETIC",
        symbols[Group1]= [               z,               Z ],
        symbols[Group2]= [     Cyrillic_ya,     Cyrillic_YA ]
    };
    key <AB02> {
        type= "ALPHABETIC",
        symbols[Group1]= [               x,               X ],
        symbols[Group2]= [    Cyrillic_che,    Cyrillic_CHE ]
    };
    key <AB03> {
        type= "ALPHABETIC",
        symbols[Group1]= [               c,               C ],
        symbols[Group2]= [     Cyrillic_es,     Cyrillic_ES ]
    };
    key <AB04> {
        type= "ALPHABETIC",
        symbols[Group1]= [               v,               V ],
        symbols[Group2]= [     Cyrillic_em,     Cyrillic_EM ]
```


© ISO/IEC 2005 – All rights reserved

```
};
key <AB05> {
    type= "ALPHABETIC",
    symbols[Group1]= [               b,               B ],
    symbols[Group2]= [      Cyrillic_i,      Cyrillic_I ]
};
key <AB06> {
    type= "ALPHABETIC",
    symbols[Group1]= [               n,               N ],
    symbols[Group2]= [     Cyrillic_te,     Cyrillic_TE ]
};
key <AB07> {
    type= "ALPHABETIC",
    symbols[Group1]= [               m,               M ],
    symbols[Group2]= [ Cyrillic_softsign, Cyrillic_SOFTSIGN ]
};
key <AB08> {
    type[group2]= "ALPHABETIC",
    symbols[Group1]= [           comma,            less ],
    symbols[Group2]= [     Cyrillic_be,     Cyrillic_BE ]
};
key <AB09> {
    type[group2]= "ALPHABETIC",
    symbols[Group1]= [          period,         greater ],
    symbols[Group2]= [     Cyrillic_yu,     Cyrillic_YU ]
};
key <AB10> {
    symbols[Group1]= [           slash,        question ],
```

© ISO/IEC 2005 – All rights reserved

```
    symbols[Group2]= [          period,           comma ]
};
key <RTSH> {          [         Shift_R ] };
key <KPMU> {
    type= "CTRL+ALT",
    symbols[Group1]= [     KP_Multiply,  XF86_ClearGrab ]
};
key <LALT> {          [           Alt_L,          Meta_L ] };
key <SPCE> {          [           space ] };
key <CAPS> {          [       Caps_Lock ] };
key <FK01> {
    type= "CTRL+ALT",
    symbols[Group1]= [              F1, XF86_Switch_VT_1 ]
};
key <FK02> {
    type= "CTRL+ALT",
    symbols[Group1]= [              F2, XF86_Switch_VT_2 ]
};
key <FK03> {
    type= "CTRL+ALT",
    symbols[Group1]= [              F3, XF86_Switch_VT_3 ]
};
key <FK04> {
    type= "CTRL+ALT",
    symbols[Group1]= [              F4, XF86_Switch_VT_4 ]
};
key <FK05> {
    type= "CTRL+ALT",
```


© ISO/IEC 2005 – All rights reserved

```
    symbols[Group1]= [              F5, XF86_Switch_VT_5 ]
};
key <FK06> {
    type= "CTRL+ALT",
    symbols[Group1]= [              F6, XF86_Switch_VT_6 ]
};
key <FK07> {
    type= "CTRL+ALT",
    symbols[Group1]= [              F7, XF86_Switch_VT_7 ]
};
key <FK08> {
    type= "CTRL+ALT",
    symbols[Group1]= [              F8, XF86_Switch_VT_8 ]
};
key <FK09> {
    type= "CTRL+ALT",
    symbols[Group1]= [              F9, XF86_Switch_VT_9 ]
};
key <FK10> {
    type= "CTRL+ALT",
    symbols[Group1]= [             F10, XF86_Switch_VT_10 ]
};
key <NMLK> {         [        Num_Lock, Pointer_EnableKeys ] };
key <SCLK> {         [     Scroll_Lock ] };
key  <KP7> {         [         KP_Home,             KP_7 ] };
key  <KP8> {         [           KP_Up,             KP_8 ] };
key  <KP9> {         [        KP_Prior,             KP_9 ] };
key <KPSU> {
```

© ISO/IEC 2005 – All rights reserved

```
    type= "CTRL+ALT",
    symbols[Group1]= [     KP_Subtract, XF86_Prev_VMode ]
};
key  <KP4> {         [         KP_Left,            KP_4 ] };
key  <KP5> {         [        KP_Begin,            KP_5 ] };
key  <KP6> {         [        KP_Right,            KP_6 ] };
key <KPAD> {
    type= "CTRL+ALT",
    symbols[Group1]= [          KP_Add, XF86_Next_VMode ]
};
key  <KP1> {         [          KP_End,            KP_1 ] };
key  <KP2> {         [         KP_Down,            KP_2 ] };
key  <KP3> {         [         KP_Next,            KP_3 ] };
key  <KP0> {         [       KP_Insert,            KP_0 ] };
key <KPDL> {
    symbols[Group1]= [       KP_Delete,      KP_Decimal ],
    symbols[Group2]= [       KP_Delete,    KP_Separator ]
};
key <MDSW> {         [             F16 ] };
key <LSGT> {
    type[group1]= "FOUR_LEVEL",
    symbols[Group1]= [            less,         greater,             bar,       brokenbar ],
    symbols[Group2]= [           slash,             bar ]
};
key <FK11> {
    type= "CTRL+ALT",
    symbols[Group1]= [             F11, XF86_Switch_VT_11 ]
};
```


© ISO/IEC 2005 – All rights reserved

```
key <FK12> {
    type= "CTRL+ALT",
    symbols[Group1]= [             F12, XF86_Switch_VT_12 ]
};
key <HOME> {         [            Home ] };
key   <UP> {         [              Up ] };
key <PGUP> {         [           Prior ] };
key <LEFT> {         [            Left ] };
key <RGHT> {         [           Right ] };
key  <END> {         [             End ] };
key <DOWN> {         [            Down ] };
key <PGDN> {         [            Next ] };
key  <INS> {         [          Insert ] };
key <DELE> {         [          Delete ] };
key <KPEN> {         [        KP_Enter ] };
key <RCTL> {         [  ISO_Next_Group ] };
key <PAUS> {
    type= "PC_BREAK",
    symbols[Group1]= [           Pause,           Break ]
};
key <PRSC> {
    type= "PC_SYSRQ",
    symbols[Group1]= [           Print,         Sys_Req,        NoSymbol ]
};
key <KPDV> {
    type= "CTRL+ALT",
    symbols[Group1]= [       KP_Divide,     XF86_Ungrab ]
};
```

© ISO/IEC 2005 – All rights reserved

```
    key <RALT> {         [           Alt_R,          Meta_R ] };
    key <LWIN> {         [         Super_L ] };
    key <RWIN> {         [         Super_R ] };
    key <MENU> {         [            Menu ] };
    key <LVL3> {         [ ISO_Level3_Shift ] };
    key  <ALT> {         [        NoSymbol,           Alt_L ] };
    key <KPEQ> {         [        KP_Equal ] };
    key <SUPR> {         [        NoSymbol,         Super_L ] };
    key <HYPR> {         [        NoSymbol,         Hyper_L ] };
    key <META> {         [        NoSymbol,          Meta_L ] };
    key  <K5D> {         [             F13 ] };
    key  <K5E> {         [             F14 ] };
    key  <K5F> {         [             F15 ] };
    key  <K6C> {         [       XF86Eject ] };
    modifier_map Control { <LCTL> };
    modifier_map Shift { <LFSH> };
    modifier_map Shift { <RTSH> };
    modifier_map Mod1 { <LALT> };
    modifier_map Lock { <CAPS> };
    modifier_map Mod2 { <NMLK> };
    modifier_map Mod5 { <MDSW> };
    modifier_map Mod5 { <LVL3> };
    modifier_map Mod1 { <ALT> };
    modifier_map Mod4 { <SUPR> };
    modifier_map Mod4 { <HYPR> };
    modifier_map Mod1 { <META> };
};
```


© ISO/IEC 2005 – All rights reserved

```
xkb_geometry "pc(pc104)" {

    width=      470;
    height=     210;

    alias <AC00> = <CAPS>;
    alias <AA00> = <LCTL>;

    baseColor=  "white";
    labelColor= "black";
    xfont=      "-*-helvetica-medium-r-normal--*-120-*-*-*-*-iso8859-1";
    description= "Generic 104";

    shape "NORM" {
        corner= 1,
        { [  18,  18 ] },
        { [   2,   1 ], [  16,  16 ] }
    };
    shape "BKSP" {
        corner= 1,
        { [  38,  18 ] },
        { [   2,   1 ], [  36,  16 ] }
    };
    shape "TABK" {
        corner= 1,
        { [  28,  18 ] },
        { [   2,   1 ], [  26,  16 ] }
    };
```

© ISO/IEC 2005 – All rights reserved

```
    shape "BKSL" {
        corner= 1,
        { [  28,  18 ] },
        { [   2,   1 ], [  26,  16 ] }
    };
    shape "RTRN" {
        corner= 1,
        { [  42,  18 ] },
        { [   2,   1 ], [  40,  16 ] }
    };
    shape "CAPS" {
        corner= 1,
        { [  33,  18 ] },
        { [   2,   1 ], [  31,  16 ] }
    };
    shape "LFSH" {
        corner= 1,
        { [  42,  18 ] },
        { [   2,   1 ], [  40,  16 ] }
    };
    shape "RTSH" {
        corner= 1,
        { [  52,  18 ] },
        { [   2,   1 ], [  50,  16 ] }
    };
    shape "MODK" {
        corner= 1,
        { [  27,  18 ] },
```


© ISO/IEC 2005 – All rights reserved

```
        { [   2,    1 ], [  25,   16 ] }
    };
    shape "SMOD" {
        corner= 1,
        { [  23,   18 ] },
        { [   2,    1 ], [  21,   16 ] }
    };
    shape "SPCE" {
        corner= 1,
        { [ 113,   18 ] },
        { [   2,    1 ], [ 111,   16 ] }
    };
    shape "KP0" {
        corner= 1,
        { [  37,   18 ] },
        { [   2,    1 ], [  35,   16 ] }
    };
    shape "KPAD" {
        corner= 1,
        { [  18,   37 ] },
        { [   2,    1 ], [  16,   35 ] }
    };
    shape "LEDS" { { [  75,   20 ] } };
    shape "LED" { { [   5,    1 ] } };
    section "Function" {
        key.color= "grey20";
        priority=  7;
        top=       52;
```

© ISO/IEC 2005 – All rights reserved

```
        left=      19;
        width=     351;
        height=    19;
        row {
            top=  1;
            left= 1;
            keys {
                {  <ESC>, "NORM",    1 },
                { <FK01>, "NORM",   20, color="white" },
                { <FK02>, "NORM",    1, color="white" },
                { <FK03>, "NORM",    1, color="white" },
                { <FK04>, "NORM",    1, color="white" },
                { <FK05>, "NORM",   11, color="white" },
                { <FK06>, "NORM",    1, color="white" },
                { <FK07>, "NORM",    1, color="white" },
                { <FK08>, "NORM",    1, color="white" },
                { <FK09>, "NORM",   11, color="white" },
                { <FK10>, "NORM",    1, color="white" },
                { <FK11>, "NORM",    1, color="white" },
                { <FK12>, "NORM",    1, color="white" },
                { <PRSC>, "NORM",    8, color="white" },
                { <SCLK>, "NORM",    1, color="white" },
                { <PAUS>, "NORM",    1, color="white" }
            };
        };
    }; // End of "Function" section

    section "Alpha" {
```


© ISO/IEC 2005 – All rights reserved

```
key.color= "white";
priority=  8;
top=       91;
left=      19;
width=     287;
height=    95;
row {
    top=  1;
    left= 1;
    keys {
        { <TLDE>, "NORM",    1 }, { <AE01>, "NORM",    1 },
        { <AE02>, "NORM",    1 }, { <AE03>, "NORM",    1 },
        { <AE04>, "NORM",    1 }, { <AE05>, "NORM",    1 },
        { <AE06>, "NORM",    1 }, { <AE07>, "NORM",    1 },
        { <AE08>, "NORM",    1 }, { <AE09>, "NORM",    1 },
        { <AE10>, "NORM",    1 }, { <AE11>, "NORM",    1 },
        { <AE12>, "NORM",    1 },
        { <BKSP>, "BKSP",    1, color="grey20" }
    };
};
row {
    top=  20;
    left= 1;
    keys {
        {  <TAB>, "TABK",    1, color="grey20" },
        { <AD01>, "NORM",    1 }, { <AD02>, "NORM",    1 },
        { <AD03>, "NORM",    1 }, { <AD04>, "NORM",    1 },
        { <AD05>, "NORM",    1 }, { <AD06>, "NORM",    1 },
```

© ISO/IEC 2005 – All rights reserved

```
        { <AD07>, "NORM",    1 }, { <AD08>, "NORM",    1 },
        { <AD09>, "NORM",    1 }, { <AD10>, "NORM",    1 },
        { <AD11>, "NORM",    1 }, { <AD12>, "NORM",    1 },
        { <BKSL>, "BKSL",    1 }
    };
};
row {
    top=  39;
    left= 1;
    keys {
        { <CAPS>, "CAPS",    1, color="grey20" },
        { <AC01>, "NORM",    1 }, { <AC02>, "NORM",    1 },
        { <AC03>, "NORM",    1 }, { <AC04>, "NORM",    1 },
        { <AC05>, "NORM",    1 }, { <AC06>, "NORM",    1 },
        { <AC07>, "NORM",    1 }, { <AC08>, "NORM",    1 },
        { <AC09>, "NORM",    1 }, { <AC10>, "NORM",    1 },
        { <AC11>, "NORM",    1 },
        { <RTRN>, "RTRN",    1, color="grey20" }
    };
};
row {
    top=  58;
    left= 1;
    keys {
        { <LFSH>, "LFSH",    1, color="grey20" },
        { <AB01>, "NORM",    1 }, { <AB02>, "NORM",    1 },
        { <AB03>, "NORM",    1 }, { <AB04>, "NORM",    1 },
        { <AB05>, "NORM",    1 }, { <AB06>, "NORM",    1 },
```


© ISO/IEC 2005 – All rights reserved

```
            { <AB07>, "NORM",    1 }, { <AB08>, "NORM",    1 },
            { <AB09>, "NORM",    1 }, { <AB10>, "NORM",    1 },
            { <RTSH>, "RTSH",    1, color="grey20" }
        };
    };
    row {
        top=  77;
        left= 1;
        keys {
            { <LCTL>, "MODK",    1, color="grey20" },
            { <LWIN>, "SMOD",    1, color="grey20" },
            { <LALT>, "SMOD",    1, color="grey20" },
            { <SPCE>, "SPCE",    1 },
            { <RALT>, "SMOD",    1, color="grey20" },
            { <RWIN>, "SMOD",    1, color="grey20" },
            { <MENU>, "SMOD",    1, color="grey20" },
            { <RCTL>, "SMOD",    1, color="grey20" }
        };
    };
}; // End of "Alpha" section

section "Editing" {
    key.color= "grey20";
    priority=  9;
    top=       91;
    left=      312;
    width=     58;
    height=    95;
```

© ISO/IEC 2005 – All rights reserved

```
row {
    top=  1;
    left= 1;
    keys {
        {  <INS>, "NORM",    1 }, { <HOME>, "NORM",    1 },
        { <PGUP>, "NORM",    1 }
    };
};
row {
    top=  20;
    left= 1;
    keys {
        { <DELE>, "NORM",    1 }, {  <END>, "NORM",    1 },
        { <PGDN>, "NORM",    1 }
    };
};
row {
    top=  58;
    left= 20;
    keys {
        {   <UP>, "NORM",    1 }
    };
};
row {
    top=  77;
    left= 1;
    keys {
        { <LEFT>, "NORM",    1 }, { <DOWN>, "NORM",    1 },
```


© ISO/IEC 2005 – All rights reserved

```
            { <RGHT>, "NORM",    1 }
        };
    };
}; // End of "Editing" section


section "Keypad" {
    key.color= "grey20";
    priority=  10;
    top=       91;
    left=      376;
    width=     77;
    height=    95;
    row {
        top=  1;
        left= 1;
        keys {
            { <NMLK>, "NORM",    1 }, { <KPDV>, "NORM",    1 },
            { <KPMU>, "NORM",    1 }, { <KPSU>, "NORM",    1 }
        };
    };
    row {
        top=  20;
        left= 1;
        keys {
            {  <KP7>, "NORM",    1, color="white" },
            {  <KP8>, "NORM",    1, color="white" },
            {  <KP9>, "NORM",    1, color="white" },
            { <KPAD>, "KPAD",    1 }
```

© ISO/IEC 2005 – All rights reserved

```
        };
    };
    row {
        top=  39;
        left= 1;
        keys {
            {  <KP4>, "NORM",    1, color="white" },
            {  <KP5>, "NORM",    1, color="white" },
            {  <KP6>, "NORM",    1, color="white" }
        };
    };
    row {
        top=  58;
        left= 1;
        keys {
            {  <KP1>, "NORM",    1, color="white" },
            {  <KP2>, "NORM",    1, color="white" },
            {  <KP3>, "NORM",    1, color="white" },
            { <KPEN>, "KPAD",    1 }
        };
    };
    row {
        top=  77;
        left= 1;
        keys {
            {  <KP0>, "KP0",    1, color="white" },
            { <KPDL>, "NORM",    1, color="white" }
        };
```


© ISO/IEC 2005 – All rights reserved

```
    };
}; // End of "Keypad" section


solid "LedPanel" {
    top=      52;
    left=     377;
    priority= 0;
    color= "grey10";
    shape= "LEDS";
};
indicator "Num Lock" {
    top=      67;
    left=     382;
    priority= 1;
    onColor= "green";
    offColor= "green30";
    shape= "LED";
};
indicator "Caps Lock" {
    top=      67;
    left=     407;
    priority= 2;
    onColor= "green";
    offColor= "green30";
    shape= "LED";
};
indicator "Scroll Lock" {
    top=      67;
```

© ISO/IEC 2005 – All rights reserved

```
        left=     433;
        priority= 3;
        onColor= "green";
        offColor= "green30";
        shape= "LED";
    };
    text "NumLockLabel" {
        top=      55;
        left=     378;
        priority= 4;
        width=  19.8;
        height=  10;
        XFont= "-*-helvetica-medium-r-normal--*-120-*-*-*-*-iso8859-1";
        text=  "Num¥nLock";
    };
    text "CapsLockLabel" {
        top=      55;
        left=     403;
        priority= 5;
        width=  26.4;
        height=  10;
        XFont= "-*-helvetica-medium-r-normal--*-120-*-*-*-*-iso8859-1";
        text=  "Caps¥nLock";
    };
    text "ScrollLockLabel" {
        top=      55;
        left=     428;
        priority= 6;
```


© ISO/IEC 2005 – All rights reserved

```
        width=  39.6;
        height=  10;
        XFont= "-*-helvetica-medium-r-normal--*-120-*-*-*-*-iso8859-1";
        text=  "Scroll¥nLock";
    };
};


};
```

© ISO/IEC 2005 – All rights reserved