ISO/IEC JTC 1/SC 35 N1052-F

DATE: 2006-10-05

ISO/IEC JTC 1/SC 35

User Interfaces

Secretariat: AFNOR

DOCUMENT TYPE: Working document

TITLE: Technologies de l'information — Modèle d'interaction des claviers — Description lisible à la machine des interactions sur clavier

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
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/CEI JTC 1/SC 35 N1052-F

Date: 2006-09-04

~~**ISO-CEI**~~

ISO/CEI JTC 1/SC 35/GT 1

Secrétariat: AFNOR

Technologies de l'information — Modèle d'interaction des claviers — Description lisible à la machine des interactions sur clavier

Information Technology — Keyboard interaction model — Key interaction model and machine-readable description of keyboard interactions

Avertissement

Ce document n'est pas une Norme internationale de l'ISO. Il est distribué pour examen et observations. Il est susceptible de modification sans préavis et ne peut être cité comme Norme internationale.

Les destinataires du présent projet sont invités à présenter, avec leurs observations, notification des droits de propriété dont ils auraient éventuellement connaissance et à fournir une documentation explicative.

Type du document:
Sous-type du document:
Stade du document:
Langue du document: F

C:\Documents and Settings\ncs\Mes documents\z62f\jeju\35n1052F-FCD-24757-20060908.doc~~D:\Donnees\SC35\ISO 24757 Description interactions clavier\ISO CEI FCD 24757 F-20060908.doc~~ STD Version 2.1c2

Notice de droit d'auteur

Sommaire

Page

Avant-propos

L'ISO (Organisation internationale de normalisation) est une fédération mondiale d'organismes nationaux de normalisation (comités membres de l'ISO). L'élaboration des Normes internationales est en général confiée aux comités techniques de l'ISO. Chaque comité membre intéressé par une étude a le droit de faire partie du comité technique créé à cet effet. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'ISO participent également aux travaux. L'ISO collabore étroitement avec la Commission électrotechnique internationale (CEI) en ce qui concerne la normalisation électrotechnique.

Les Normes internationales sont rédigées conformément aux règles données dans les Directives ISO/CEI, Partie 2.

La tâche principale des comités techniques est d'élaborer les Normes internationales. Les projets de Normes internationales adoptés par les comités techniques sont soumis aux comités membres pour vote. Leur publication comme Normes internationales requiert l'approbation de 75 % au moins des comités membres votants.

L'attention est appelée sur le fait que certains des éléments du présent document peuvent faire l'objet de droits de propriété intellectuelle ou de droits analogues. L'ISO ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de propriété et averti de leur existence.

L'ISO/CEI xxxxx a été élaborée par le comité technique ISO/TC JTC 1, *Technologies de l'information*, sous-comité SC 35, *Interfaces utilisateur*.

Introduction

La présente norme internationale s'adresse à ceux qui conçoivent des systèmes d'exploitation et des applications logicielles prenant le clavier en charge (y compris pour la présentation complète du clavier à l'écran à des fins de documentation). Son but est d' harmoniser les pratiques de l'industrie en matière de description machine des claviers (PC, assistants numériques, Linux, Windows, Apple, …) Elle vise finalement à faciliter la production de pilotes interopérables pour les utilisateurs et à permettre une meilleure assistance à l'utilisateur en offrant une correspondance plus précise entre le clavier matériel tel que gravé et géométriquement configuré, et l'interface conceptuelle dont dispose le système d'exploitation et ses applications.

Technologies de l'information — Modèle d'interaction des claviers — Description lisible à la machine des interactions sur clavier

1 Domaine d'application

La présente norme internationale a pour but de procurer un format de description qui peut non seulement décrire entièrement les normes internationales de clavier mais aussi décrire les possibilités des claviers du marché actuel et envisageable dans un avenir prévisible, de même que leur fonctionnement avec des systèmes d'exploitation correspondants. Elle décrit les interactions possibles entre les touches d'un clavier et normalise la description des claviers pour qu'elle soit lisible à la machine tout en restant interprétable par des humains de manière relativement facile.

2 Conformité

La description lisible à la machine d'un clavier est conforme à la présente norme internationale si les prescriptions des articles 5.x à 5.y sont respectées.

3 Références normatives

Les documents de référence suivants sont indispensables pour l'application du présent document. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

ISO 639-2 :1998, *Codes pour la représentation des noms de langue -- Partie 2: Code alpha-3*

ISO 3166-1:1997, *Codes pour la représentation des noms de pays et de leurs subdivisions -- Partie 1: Codes pays*

ISO/CEI 9995-1:2006, *Technologies de l'information — Disposition des claviers conçus pour la bureautique — Partie 1: Principes généraux pour la disposition des claviers*

ISO/CEI 10646:2003, *Technologies de l'information — Jeu universel de caractères codés sur plusieurs octets (JUC)*

ISO/IEC 19757-2:2003 *Technologies de l'information -- Langage de définition de schéma de documents (DSDL) -- Partie 2: Validation de grammaire orientée courante -- RELAX NG*

4 Termes et définitions

Pour les besoins du présent document, les termes et définitions donnés dans l'ISO/CEI 9995-1 s'appliquent. Les définitions supplémentaires suivantes s'appliquent aussi.

4.1
coordonnées de référence
identifiant formé d'une lettre et de deux chiffres décimaux, se référant à la grille de numérotation permettant de positionner une touche à l'intersection d'une rangée et d'une colonne du clavier, où les rangées sont identifiées par des lettres et les colonnes par des chiffres

5 Spécifications

5.1 Description des touches et des dispositions de clavier

Le clavier est décrits pour des machines dans la présente norme internationale en utilisant les propriétés suivantes :

- les coordonnées logiques de chaque touche suivant un système de numérotation par grille établi dans l'ISO/CEI 9995-1. Ces coordonnées sont appelées « coordonnées de référence » ;

  Note: Les coordonnées de référence d'un clavier peuvent correspondre à une norme de clavier.

- la description de la taille en millimètres (mm), de la forme et des coordonnées physiques de chaque touche d'un clavier matériel, pour permettre de reproduire précisément le clavier de l' utilisateur à l'écran ;

- optionnellement, région ou pays principal et code de langue applicables à cette disposition ;

- l'apparence réelle de l'étiquetage ;

- la division logique de chaque touche du clavier en groupes et en niveaux au sein de chaque groupe.

Au sein d'un groupe, à un niveau donné, chaque caractère est décrit en utilisant cinq autres propriétés :

- par un identifiant formé de la lettre U et du numéro qu'il occupe dans le jeu universel de caractères (ISO/CEI 10646) ; si le caractère est composé ou obtenu par combinaison, les identifiants des caractères individuels qui composent le caractère résultant sont donnés avec la séquence qui permet cette composition ;

- optionnellement, par son nom de caractère dans une langue donnée, identifiée selon son codet dans l'ISO 639-1 ou selon son codet terminologique dans l'ISO/CEI 639-2 ;

- optionnellement, par un ou plusieurs « codes de balayage » (qui peut varier selon l'état du clavier, selon le matériel utilisé, etc.) ;

- dans le cas d'une position qui fait de la touche une touché morte, une description des combinaisons requises par l'utilisateur, avec le ou les caractères qui en résultent ;

- optionnellement, une étiquette pour identifier les caractères dont le glyphe gravé sur la touche est différent de sa représentation interne (comme par exemple, U+00A6 barre verticale discontinue par opposition à U+007C barre verticale).

Note: La description des combinations requises n'interdit pas aux pilotes de clavier de prendre en compte d'autres combinaisons pour créer d'autres caractères si ces combinaisons sont déjà définies. La description don't il est ici question est destine à assurer que les besoins de l'utilisateur sont satisfaits, quelles que soient les définitions préalables.

Cette description ne peut faire abstraction de l'interaction des touches entre elles pour produire des résultats.

L'article qui suit décrit les différents états d'un clavier nécessaires à la saisie des caractères et pris en compte dans la présente norme internationale.

5.2 États produits par l'interaction des touches de fonction utilisées pour la saisie

La frappe d'une des touches de fonction suivantes détermine un état produisant le caractère désiré. Cet article normalise les états conventionnels produits par l'interaction de ces touches avec des touches alphanumériques.

- Sélection du niveau 2 (appelé aussi MAJ, en anglais « Shift ») : cette touche en soi permet la saisie de caractères de niveau 2 dans le groupe actif.

- Sélection du niveau 3 (appelé aussi AltCar) : cette touche en soi permet la saisie de caractères de niveau 3 dans le groupe actif.

- Contrôle : cette touche est souvent utilisée en pratique avec une des touches précédentes pour la saisie de caractères. Voir ci-après l'interprétation que prescrit la présente norme internationale pour ces interactions.

- Sélection de groupe : cette touche en soi permet le changement de groupe. Cette action peut être verrouillante ou non.

  (Cette description dans une annexe – généraliser la sélection de groupe)

  L'ISO/CEI 9995-2 précise ce qui suit à cet effet :

  Pour la saisie du répertoire de caractères graphiques de la collection 281 (intitulée MES-1), précisée dans l'amendement 1 à l'ISO/CEI 10646:1-2000, l'on spécifie dans l'ISO/CEI 9995-3 la disposition du *groupe secondaire courant* (qui doit être utilisé en tant que groupe 2). Particulièrement pour le groupe 2, l'on recommande que la fonction de *sélection de groupe* verrouille ce groupe pour la saisie du prochain caractère, et pour ce caractère seulement. Autrement dit, l'activation du groupe 2 change l'état logiciel du clavier de telle sorte que même si l'on relâche toutes les touches impliquées dans le processus d'activation, la prochaine fois que l'on appuie sur une touche, celle-ci sélectionnera un caractère du groupe 2. Après avoir saisi un tel caractère dans ce mode, le clavier revient automatiquement au groupe qui prévalait avant que l'on passe au groupe 2.

  NOTE On recommande, lors de la sélection d'un groupe qui couvre un système d'écriture complet (par exemple l'hiragana, le katakana, le cyrillique, le grec, l'arabe, l'hébreu), de verrouiller ce groupe à demeure jusqu'à ce qu'on le désactive ou que l'on sélectionne un autre groupe (à titre d'exemple, après avoir activé le groupe hiragana, il est d'usage de le désactiver explicitement pour retourner au groupe 1). La manière exacte d'activer la sélection de groupe avec une fonction de *sélection de groupe* n'est pour le moment pas normalisée. Il est recommandé comme mesure minimale d'indiquer visuellement tout verrouillage de groupe par un moyen approprié (par exemple une lampe, un affichage à cristaux liquides ou une indication à l'écran, sauf pour les groupes 1 et 2.

- Combinaisons habituelles de certaines touches de fonction et leur interprétation normalisée :

  a) *Sélection de niveau 2* + *Sélection de niveau 3* (MAJ + AltCar) doit s'interpréter comme suit (deux scénarios sont possibles) :

  1. Selon l'ISO/CEI 9995-2 : « Particulièrement, pour l'agencement de clavier harmonisé à 48 touches graphiques, si des caractères sont attribués à plus d'un groupe, la fonction de *sélection de groupe* doit s'activer quand on appuie sur une touche de *sélection du niveau 3 [AltCar]* alors que l'on appuie déjà sur une touche de *sélection du niveau 2 [MAJ]*, ou vice-versa. » On comprendra que le relâchement de ces deux touches sans frapper entre-temps une autre touche doit avoir le même effet que frapper une touche de changement de groupe dédiée.

  2. Si une touche alphanumérique est frappée pendant que les touches *Sélection de niveau 2* et *Sélection de niveau 3* sont enfoncées, l'état du clavier doit être interprété comme donnant accès à un groupe apparenté au groupe précédemment actif, mais différent, au niveau 1. Certaines implantations considèrent ceci comme équivalent à un niveau 4 virtuel dans le groupe actif. Même si cela ne correspond pas à un

concept normalisé, cette vue est tolérable dans le contexte d'un cercle linguistique restreint.

b) *Sélection de niveau 2 [MAJ]* + *Contrôle* doit s'interpréter comme suit :

Si une touche alphanumérique est frappée pendant que les touches *Sélection de niveau 2 [MAJ]* et *Contrôle* sont enfoncées, l'état du clavier doit être interprété comme se trouvant dans un groupe apparenté au groupe actif, mais différent, au niveau 2. Certaines interprétations considèrent ceci comme équivalent à un niveau 5 virtuel dans le groupe actif. Même si cela ne correspond pas à un concept normalisé, cette vue est tolérable dans le contexte d'un cercle linguistique restreint.

c) *Sélection de niveau 3 [AltCar]* + *Contrôle* doit s'interpréter comme suit :

Si une touche alphanumérique est frappée pendant que les touches *Sélection de niveau 3* et *Contrôle* sont enfoncées, l'état du clavier doit être interprété comme se trouvant dans un groupe apparenté au groupe actif, mais différent, au niveau 3. Certaines interprétations considèrent ceci comme équivalent à un niveau 6 virtuel dans le groupe actif. Même si cela ne correspond pas à un concept normalisé, cette vue est tolérable dans le contexte d'un cercle linguistique restreint.

d) *Sélection de niveau 2 [MAJ]* + *Sélection de groupe* doit s'interpréter comme suit :

Si une touche alphanumérique est frappée pendant que les touches *Sélection de niveau 2* et *Sélection de groupe* sont enfoncées, l'état du clavier doit être interprété comme se trouvant dans un groupe apparenté au groupe actif, mais différent, au niveau 2. Certaines interprétations considèrent ceci comme équivalent à un niveau 5 virtuel dans le groupe actif. Même si cela ne correspond pas à un concept normalisé, cette vue est tolérable dans le contexte d'un cercle linguistique donné. Par exemple, dans le contexte de l'ISO/CEI 9995-3, où le groupe 1 est un groupe latin national et où le groupe apparenté est le groupe 2 (« groupe secondaire courant » selon l'appellation de l'ISO/CEI 9995-3), l'état du clavier situe la saisie du prochain caractère dans le groupe 2, au niveau 2.

e) *Sélection de niveau 3 [AltCar]* + *Sélection de groupe* doit s'interpréter comme suit :

Si une touche alphanumérique est frappée pendant que les touches *Sélection de niveau 3* et *Sélection de groupe* sont enfoncées, l'état du clavier doit être interprété comme se trouvant dans un groupe apparenté au groupe actif, mais différent, au niveau 3. Certaines interprétations considèrent ceci comme équivalent à un niveau 6 virtuel dans le groupe actif. Même si cela ne correspond pas à un concept normalisé, cette vue est tolérable dans le contexte d'un cercle linguistique donné. Par exemple, dans le contexte de l'ISO/CEI 9995-3, où le groupe 1 est un groupe latin national et où le groupe apparenté est le groupe 2 (« groupe secondaire courant » selon l'appellation de l'ISO/CEI 9995-3), l'état du clavier se situe dans le groupe 2, au niveau 3.

5.3 Touches spéciales

Les touches de fonction suivantes existent sur différents claviers commercialisés. Le type peut en être généralisé au besoin.

a) Au Japon, les textes peuvent être rédigés en écriture phonétique (kanas) ou en écriture kanji (caractères chinois). En Corée on a recours à une méthode semblable pour convertir l'hangoûl (lettres coréennes) en hanza (caractères chinois). Une touche de conversion kana-kanji (ou hangoûl-hanza) permet de convertir à la volée une suite de caractères phonétiques déjà saisis en caractères chinois. Cette touche ne sert pas à la saisie de caractères en combinaison avec d'autres touches mais doit être décrite comme touche de fonction spéciale qui fait entrer en jeu un programme de transformation de caractères déjà saisis.

b) Sur certains claviers hors-Japon ou hors-Corée, il existe une touche combinatoire qui joue un peu le même rôle que la touche kana-kanji mais généralisée à la création de caractères non disponibles au clavier à partir de caractères déjà saisis sur celui-là (exemple fictif : la création du caractère § à partir de l'amalgame des deux caractères s et s, ou S et S, ou toute combinaison de ces deux caractères). Le fonctionnement de cette touche combinatoire n'est pas normalisé pour l'instant. Sur certains claviers, la composition est transitive tandis que sur d'autres elle est plus statique. Comme pour la touche kana-kanji, la frappe de la touche combinatoire fait entrer en jeu un programme de transformation de caractères déjà saisis..

5.4 Touches mortes

Certaines touches sont dites mortes en cela que leur frappe crée un caractère partiel qui peut ou non apparaître à l'écran. Le caractère complet (dit aussi « pleinement formé » ou « précomposé ») est en général formé d'une lettre de base et d'un ou de plusieurs signes diacritiques.

NOTE La norme ISO/CEI 9995-3 dit de la formation de caractères pleinement formés à l'aide de touches mortes ce qui suit :

« *Les signes diacritiques figurent au-dessus ou au-dessous de certaines lettres, et sont tous des caractères affectés à une touche morte. Enfoncer une touche représentant un signe diacritique, puis enfoncer une touche représentant une lettre, doit indiquer que les symboles graphiques de deux caractères sont destinés à être combinés. Enfoncer une touche avec un signe diacritique puis appuyer sur la barre d'espace, doit indiquer que le signe diacritique est destiné à apparaître comme un caractère graphique à part entière (c'est-à-dire physiquement autonome).*

Il est recommandé que la méthode utilisée pour l'effacement d'un caractère soit également utilisée pour annuler un caractère incomplètement saisi, tel qu'un signe diacritique non suivi de la lettre ou de l'espace auquel il devrait être combiné. »

Note: Les signes diacritiques peuvent aussi figurer partout autour du cors des lettres et même à l'intérieur.

À cela on doit ajouter que l'usage de touches mortes peut aussi être transitif, comme dans le cas d'une touche combinatoire, c'est-à-dire que certaines écritures (grec polytonique, vietnamien) prévoient parfois plusieurs diacritiques sur une même lettre de base. À moins que les lettres de base ne soient disponible directement au clavier sous forme de caractères précomposés avec un premier signe diacritique intégré (comme dans le cas de la lettre scandinave å, qui peut aussi être affectée d'autres accents), il est alors nécessaire que la fonctionnalité des signes diacritiques soit transitive, c'est-à-dire que la saisie de plusieurs signes diacritiques de suite puisse s'applique au caractère de base dont la frappe vient en dernier.

5.5 Rémanence et verrouillage stable (explicite) ou temporaire (implicite)

L'état d'un clavier peut être verrouillé de façon stable ou temporaire, généralement après activation par une touche de fonction ou une séquence de touches.

5.5.1 Verrouillage stable (explicite)

Les touches ou fonctions suivantes provoquent un verrouillage stable, qui n'est désactivé qu'en appuyant à nouveau sur la même touche ou en appelant à nouveau la même fonction :

- Blocage en majuscules
- Blocage au niveau 2
- Blocage de groupe
- Blocage numérique

5.5.2 Verrouillage temporaire (implicite)

La norme ISO/CEI 9995-3 recommande, lorsque l'on appuie sur la touche de sélection de groupe pour appeler le groupe 2 (appelé *groupe secondaire courant*), le verrouillage de l'état du clavier pour la frappe de la prochaine touche seulement (voir aussi 5.2 sous *Sélection de groupe*). En fait ce comportement dépend de la

nature du groupe appelé, et pourrait s'appliquer à d'autres groupes que le groupe 2. Le groupe 2 comporte des touches mortes dont les accents s'appliquent à des caractères latins de base qui se trouvent normalement dans le groupe qui prévalait avant l'appel du groupe 2. Il est logique que l'on bascule au groupe précédent pour la frappe de cette touche. Les autres caractères du groupe 2 sont aussi des caractères peu fréquents.

Un autre groupe qui comporterait des caractères peu fréquents peut avoir comme propriété ce verrouillage temporaire. La définition du groupe doit donc prévoir cette propriété de verrouillage temporaire implicite (autrement le groupe doit être verrouillé explicitement).

5.5.3 Rémanence (accessibilité)

La rémanence est une forme de verrouillage temporaire de fonction qui s'applique aux touches de fonction normalement utilisées en interaction avec d'autres touches. L'usage premier est de permettre aux handicapés de frapper des touches une par une, sans que l'on doive appuyer sur deux touches à la fois (par exemple pour la saisie de la majuscule initiale d'une phrase, le fait de frapper la touche de Sélection de niveau 2 dans ce mode implicite bloquera l'état du clavier en majuscules pour la frappe de la prochaine touche seulement).

En mode de rémanence, le fait d'appuyer successivement sur des touches de fonction, une à une, et de la relâcher, est équivalent à maintenir virtuellement une pression sur toutes ces touches à la fois tant que l'on n'a pas frappé une touche alphanumérique. À ce moment seulement, le maintien de ces touches de fonction est virtuellement désactivé.

Le mode de rémanence est activé en appuyant cinq fois de suite sur une touche de sélection du niveau 2. Il est désactivé en appuyant à nouveau cinq fois de suite sur une touche de niveau 2.

5.6 Touches supplémentaires (accessibles ou non par le logiciel de l'ordinateur) ; Courriel, touches de fonctions Fn, touches « Windows », Sommeil, Départ/Arrêt)

Même si ces touches envoient un code de balayage déjà traité par l'ordinateur, elles devraient être prises en compte comme toutes les autres touches, car techniquement elles ne font pas exception à ce que la description de clavier pourrait contenir.

Les autres touches (celles qui n'envoient pas de code de balayage à l'ordinateur) peuvent être décrites textuellement, puisque le logiciel peut avoir besoin de connaître leur existence et leur emplacement géométrique sur le clavier pour être en mesure de venir en aide à l'utilisateur.

[Veut-on décrire ces touches – en particulier celles sans code de balayage – de sorte que le clavier au complet soit reproduisible fidèlement sur écran pour l'utilisateur ?]

Réponse : OUI

5.7 Changement d'état complexe (principalement pour l'accessibilité)

Cet article normalise les changements d'état suivants :

Appui successif de la touche de sélection du niveau 2 cinq fois :

Appui de la touche de sélection du niveau 2 pendant 10 secondes :

[Alt][Sélection du niveau 2][Impression de l'écran] :

etc.

[Contributions sollicitées des organismes nationaux concernant la normalisation de cet aspect]

5.8 Rétroaction – données sur le modèle du clavier

Un protocole pour les systèmes d'exploitation est décrit à l'annexe A pour l'interrogation du matériel sur le modèle précis de clavier utilisé. La réponse du clavier est sous forme de fichier respectant le format décrit à l'article 5.9. Cette fonctionnalité est optionnelle dans la présente normeinternationale et elle implante l'approche « prêt à l'utilisation » pour les claviers.

5.9 Langage de description de clavier lisible à la machine

Le format de description de clavier est destiné à rendre possible la description des capacités des claviers matériels et des logiciels qui leur sont associés, plus des extensions prévisibles. Il est donc souhaitable de definer ce format dans un langage international extensible comme le langage SGML de l'ISO, dans la forme connue sous le nom de *RELAX NG de l'ISO*, aisément convertible au standard de l'industrie XML. Le format est descrit à l'Annexe B.

Le format de description de clavier est destiné au départ à l'usage des systèmes d'exploitation, et lors de son processus d'amorçage (par exemple dans le BIOS), mais il peut aussi être utilisé à d'autres fins, telles que la communication automatique du materiel au logiciel pour aider le système d'exploitation à configurer le pilote de clavier, ou pour présenter à l'écran le clavier à l'aide d'une image conviviale.

Un bon test permettant de constater su ce format est adapté aux logiciels existants consiste à lui faire correspondre les formats existants sur le marché, comme les descriptions de clavier de Microsoft, les définitions de clavier « X keyboard », les définitions de clavier Unix en mode « lignes de commande » et celles définies en langage XML. Une description de différents standards de l'industrie pour la definition formelle des claviers est donnée à l'annexe D avec leur correspondance dans le format prescrit par la prsente norme internationale.

Des fonctionnalités trouvées sur quelques produits sont couverts, comme les claviers avec touches programmables et les claviers avec attributions multiples tels que les pavés téléphoniques.

Le format de description de clavier est décrit en 4 sections:

1. une section sur l'identification du clavier et ses caractéristiques générales, incluant la marque et le modèle, le numéro de série, le pays ou la région et la langue auxquels le clavier s'applique, l'identification de la langue de gravure, et certaines particularités distinctives, comme le relief ou la présence de voyants lumineux sur les touches.

2. la disposition géométrique matérielle, qui indique une disposition géométrique généralement reconnue telle que celle d'un clavier à 102 touches de PC. Cette section renseigne aussi sur les caractéristiques physiques du clavier, comme la taille des touches, et la force de pression requise pour activer les touches.

3. la disposition de clavier, qui donne l'attribution exacte des caractères aux touches..

4. les combinaisons de touches, qui donnent les combinaisons telles que celles les caractères affectés les touches mortes.

Chacune de ces données peut être omises et le système d'exploitation ou l'utilisateur peuvent les modifier en fonction de leurs préférences.

L'ordre de priorité d'application des modifications precedence place l'utilisateur en premier, le système d'exploitation en second et la description en provenance du matériel en dernier.

Dans l'annexe D, quelques formats de définitions de claviers sont donnés, avec une correspondance entre ces formats et celui prescrit par la présente norme internationale.

Annexe A (normative) – Protocole pour l'échange d'information entre le clavier matériel et le logiciel

Cette annexe définit un protocole qui fait en sorte qu'un clavier conforme rapporte sa configuration au système d'exploitation.

La commande demandant au clavier le rapport sur sa configuration est placée oar le système d'exploitation en envoyant au clavier la séquence auivante :

Activer le mode de blocage en majuscules
Activer le mode de blocage numérique
Désactiver le mode de blocage numérique

Activer le mode de blocage numérique
Désactiver le mode de blocage en majuscules
Désactiver le mode de blocage numérique

Note: Voir l'annexe C pour une description de quelques différents stades dans l'histoire des claviers d'ordinateurs et de leurs logiciels associés.

Annexe B (normative) – Langage de description des claviers

Cette annexe précise la sémantique et la syntaxe du format de description des claviers prescrit par la présente norme internationale.

B.1 Description des formats

À produire

B2. Spécification en langage RELAX NG du format de description des claviers

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

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


# Référence à un ensemble de modificateurs

setOfModifiers =

  element modifier {

    attribute name { xsd:string },

    empty

  }+

# Quelques actions affectent soit le groupe ou

# les modificateurs

stateChangeAction =

  (# Group management.

   # "relative" attribute

   # indicates whether

   # the value should be

   # treated as absolute or

   # added/removed to the

   # current value

   attribute type { "group" },

   attribute relative { xsd:boolean },
```

```
   xsd:integer)
  | (# La liste des modificateurs pour
     # set/latch/lock (activer/enclencher/verrouiller)
     attribute type { "modifiers" },
     setOfModifiers)
# Comment les indicateurs utilisent les groupes ou les modificateurs
indicatorUse =
  element use {
    attribute which {
      "base" | "compat" | "effective" | "latched" | "locked"
    },
    empty
  }+
customProperties =
  # Tout autre paramètre – propriétés taillées sur mesure, si necessaire.
  # Peut être utilisé dans les pilotes ou des applications spécifiques à une
marque.
  element customProperty {
    attribute name { xsd:string },
    xsd:string
  }*
start =
  element keyboard {
    # First, all the dictionaries we use in the definitions
    element dictionaries {
      # Liste des codets (keycodes).
      # La liste par défaut devrait être produite séparément
      element keycodes {
      # Chaque codet (keycode) est representé par un identifiant (chaîne)
      # et une valeur numérique
```

```
  element keycode {
    (attribute id { xsd:string }
     & attribute value { xsd:int }),
    empty
  }+,
  # Liste des alias
  element alias {
    (attribute id { xsd:string }
     & attribute ref { xsd:string }),
    empty
  }*
},
# Liste des modificateurs utilisés dans la définition
element modifiers {
  # Chaque modificateur n'a rien d'autre qu'un nom
  element modifier {
    attribute name { xsd:string },
    empty
  }*
},
# Types de clavier
element types {
  element type {
    (# Chaque type a un nom
     attribute name { xsd:string }
     & # Nombre de niveaux pour un type
       attribute numLevels { xsd:int }),
    # Correspondance (map): (modificateurs)->niveau
    element map {
      element mapEntry {
```

```
              attribute level { xsd:int },
              # Ensemble de modificateurs (peut être vide)
              # Ils doivent faire référence aux éléments du dictionnaire des
modificateurs
              element modifier {
                attribute name { xsd:string },
                empty
              }*
            }*
          }
        }*
      },
      # Règles d'interprétation
      element interpretation {
        element rule {
          # Chaque règle comporte une condition d'exécution
          element condition {
            # Codet (keycode) pour lequel la règle s'applique
            element keysym {
              attribute id { xsd:string },
              empty
            },
            # Liste des modificateurs (pour l'attribut predicate [prédicat]).
            element modifiersList {
              attribute predicate {
                "all" | "any" | "exactly" | "none"
              }?,
              element modifier {
                attribute name { xsd:string },
                empty
```

```
          }+
        }?
      },
      # Quand la règle s'applique : ce que le système doit faire
      element action {
        # L'action consiste à ...
        element latch { stateChangeAction }
        | element lock { stateChangeAction }
        | element set { stateChangeAction }
      }
    }*
  }
},
# Méta-information – décrit l'usage du clavier
element meta {
  # Liste de pays, peut être vide. ISO 3166
  element countriesList {
    element country { xsd:string }*
  },
  # Liste de langues, peut être vide. ISO 639
  element languagesList {
    element language { xsd:string }*
  },
  customProperties
},
element nonFunctional {
  element geometry { external "Tiny-1.2.rnc" }
},
element functional {
  element groupList {
```

```
  element group {
    attribute name { xsd:string }
  }+
},
# Liste des touches du clavier
element keyList {
  element key {
    (# Physical scancode
     attribute scancode { xsd:int }
     & # One keycode per key -
       # see the keycodes dictionary above
       attribute keycode { xsd:string }),
    # Méta-information sur la touche (optionnelle)
    element meta {
      # Gravure (image matricielle)
      element engraving {
        attribute format { "png" | "gif" | "bmp" },
        xsd:base64Binary
      }?,
      customProperties
    }?,
    # Chaque touche peut comporter de 0 à N groupes
    element group {
      # Pour chaque groupe, une touche comporte un type
      attribute type { xsd:string },
      # Pour chaque groupe, il peut y avoir plusieurs niveaux de sélection
      # Sur chaque niveau, on peut trouver soit un symbole,
      #  soit une séquence
      element shiftLevel {
        element symbol { xsd:string }+
```

```
            }+
          }*
        }+
      },
      # Liste des indicateurs
      element indicatorsList {
        element indicator {
          # Chaque indicateur réel devrait avoir un nom
          attribute name { xsd:string },
          element groups {
            indicatorUse
            # Quels groupes sont reflétés par ce modificateur

          }?,
          element modifiers {
            indicatorUse,
            # Quels modificateurs sont reflétés par ce modificateur
            setOfModifiers
          }?
        }+
      }
    }
  }
```

B.3 Description en langage EBNF du format de clavier

À produire

B.4 Un exemple de description de clavier

À produire

Annexe C (informative) – Brève histoire des claviers d'ordinateurs et des logiciels associés

L'évolution technique des claviers de PC s'est déroulée en phases successives, dans l'ordre suivant : le clavier du PC initial d'IBM, le clavier du PC AT et l'interface USB. Dans la description suivante, les possibilités techniques de chacune de ces phases sont présentées.

A.1 Le clavier IBM

Le clavier IBM est apparu avec le PC initial d'IBM en 1981. Plusieurs claviers d'ordinateur existaient bien sûr avant cela.

Un petit processeur a alors été inclus dans le clavier, qui surveillait l'activité des touches et envoyait en conséquence des codes à l'ordinateur. Les codes (appelés scancodes en anglais) représentaient alors une valeur comprise entre 1 et 127. De plus un code d'état était envoyé pour toute frappe, soit un code pour indiquer le début de la frappe et un autre pour en marquer la fin. Le processeur était aussi responsable d'envoyer des codes à répétition, lorsqu'une touché était enfoncée pendant une durée prolongée. Le code d'état (début ou fin de frappe) utilisait un bit au début du code de balayage à un seul octet trsnamis à l'ordinateur.

Le processeur du clavier pouvait aussi recevoir un ordre de l'ordinateur pour modifier l'état de trois modes de verrouillage, le verrouillage numérique, le blocage en capitales, et le blocage du défilement. Ce codeallumait ausi un voyant sur le clavier pour donner une indication de ces états à l'utilisateur.

L'interface entre l'ordinateur et le clavier consistait en une interface série DB9 à 9 broches, ou une fiche plus grande à 5 broches. Les deux se conformaient à l'interface V.24, et l'on pouvait trouver des câbles et des fiches qui agissaient comme convertisseurs mécaniques d'une interface à une autre, simplement de broche à broche. Le taux de transmission de l'interface était généralement de 9600 bits par seconde.

La disposition matérielle du clavier du PC d' IBM consistait en une rangée de 10 touches de fonction sur le côté gauche, d'une section principale avec un nombre variable de touches, selon le marché desservi (pays, langue). Le clavier du PC d'origine comportait un seul module pour l'édition et le pavé numérique, et la façon de basculer entre les deux consistait à utiliser la touche de blocage numérique. Il n'y avait pas de touches de fonction ou de touches spéciales dans la partie supérieure du clavier.

A.1 Le clavier du PC AT et du PS/2

Le clavier du PC AT était semblable au clavier du PC initial. Il communiquait aussi des codes de balayageet d'état à l'ordinateur et recevait des changements d'état de celui-là.

La difference se trouvait surtout dans la disposition du clavier, alors que les 10 touches de fonction à gauche avaient été déplacées dans la rangée supérieure du clavier, avec l'ajout de deux touches de plus (12 touches de fonction). De plus, un module d'étition indépendant a été ajouté alors que le module numériquevoyait sa compatilibilté avec l'ancien PC préservée (il était toujours possible de basculer d'un module numérique au pavé numérique en utilisant la touche de blocage numérique).

Les codes de balayage et d'état étaient alors transmis sur deux octets plutôt que sur un octet les combinant.

La fiche du clavier de PS/2 était plus petite, avec 6 broches, même si elle restait électriquement compatible avec la fiche DB9 et celle à 5 broches du PC d'origine. La fiche du PS/2 était souvent de couleur bleue, et comportait un côté plat, ou autrement marqué sur le dessus. Des câbles de conversion entre les différents formats de fiche existaient et étaient couramment utilisés.

(images du clavier de l'AT et d'une fiche de PS/2).

A.1 Le clavier USB

Le clavier USB a introduit une nouvelle generation de composantes électroniques pour cette unité. La disposition matérielle du clavier était semblable à celle du clavier de l'AT, avec des modifications minimes, mais ce qui s'échangeait entre le clavier et l'ordinateur était différent. Les valeurs des codes de touche pouvaient désormais aller bien au-delà de 127. Par ailleurs plus d'information concernant le clavier pouvait être obtenue via l'interface USB.. D'une certaine manière, l'interface devait aussi pouvoir être électriquement compatible et aussi compatible pour ce qui est des codes, puisqu'il existe des fiches qui convertissent mécaniquement les signaux entre les interfaces USB et PS/2.

Annexe D (informative) – Correspondance entre les formats

Cette annexe recense différents formats de description de claviers interprétables par des machine et qui sont utilises par différents systèmes d'exploitation.

Pour chaque format de description, un aperçu du format est présenté, suivi d'un exemple, puis une description en langage BNF et finalement une correspondance entre ce format et celui prescrit par la présente norme internationale. L'information donnée se veut complete, sauf pour les erreurs qui se seraient glissées dans la description.

E.1 Pilotes de claviers de Microsoft

À produire

Le texte comprendra un exemple produit par le programme MKLC, distribué gracieusement à l'adresse : http://www.microsoft.com/globaldev/tools/msklc.mspx

E.2 Format xkb utilisé dans XWindow sous Unix/Linux

Le format xkb fait partie du système graphiquie XWindow.

E.2.1 Descriptions xkb

À produire

E.2.2 Exemple de xkb

Cet exemple est donné à titre de demonstration, et ne constitue pas une spécification de production. Il a été produit pour la présente norme internationale. L'exemple vise la demonstration de toutes les possibilities de xkb..

```
xkb_keymap {

xkb_keycodes "xfree86+aliases(qwerty)" {

    minimum = 8;

    maximum = 255;

     <ESC> = 9;

    <AE01> = 10;

    <AE02> = 11;

    <AE03> = 12;

    <AE04> = 13;
```

<AE05> = 14;

<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;

<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;

<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;
<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;
<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;

<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;

<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;

<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;
    <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>;
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>;
    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;
        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";
```

```
};
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";
    };
    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;
        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;
        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";
```

```
};
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";
```

```
};
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";
```

```
    };
};

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);
```

```
};
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;
        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) {
    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) {
        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;
        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;
        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;
        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;
        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;
        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) {
        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);
};
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);
```

```
    };
    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) {
    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;
        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;

        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);
    };
    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;
    };
    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 ],
        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",
    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 ]
    };
    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 ] };
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 ],
        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> {
    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 ]
};
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 ],
    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",
        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> {
    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 ]
```

```
};
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 ]
```

```
    };
    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> };
};
```

```
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 ] }
```

```
};
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 ] },
        { [   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;
    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" {
        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 },
        { <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 },
            { <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;
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 },
            { <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 }
    };
};
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" }
```

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


};
```