5i’

SECTION 3

EXTENDED MML FOR VISUAL DISPLAY TERMINALS

Recommendation Z.321

INTRODUCTION TO THE EXTENDED MML

FOR |
VISUAL DISPLAY TERMINALS

1 Scope of the Section

This Section deals with man-machine interfaces that take advantage of the input and output facilities usually available on visual display terminals (VDTs). The procedures described are not necessarily confined to this type of terminal; they can also be applied to printer-oriented terminals, such as teletypewriters, within the limits imposed by the facilities available at those terminals, e.g., information entry through menu selection.

By maintaining consistency with Recommendations Z.311-Z.317, these Recommendations facilitate a transition from a man-machine interface using basic syntax and dialogue procedures as described in Section 1 to one based on VDTs.

Diagrams and examples are used to clarify and illustrate the concepts explained in the text. The diagrams do not include exceptional cases and do not specify all possibilities available with the extended MML; those not shown diagrammatically, but which are allowed in the text, are subjects for further study and are not excluded from the extended MML. Similarly, the examples shown are not intended to imply a particular system implementation.

The Recommendations cover aspects of VDTs that users see and use, e.g., data entry, data display, interactive control, user guidance, etc. Specific terminal characteristics are avoided wherever possible.

2 Organization of Section 3

Section 3 consists of the following Recommendations:

Z.321 Introduction to the extended MML for visual display terminals

Z.322 Capabilities of visual display terminals

Z.323 Man-machine interaction

Recommendation Z.322 | escribes many of the capabilities currently available in VDTs. *Recommendation Z.323* focusses on actual man-machine interactions (i.e., *how* the capabilities are used) by addressing various aspects such as dialogue elements, monologue outputs, user assistance and interactive control.

3 Human factors

3.1 *The human factor view of the man-machine interface*

Human factor science characterizes the man-machine interface as any part of a system that the user comes in contact with — either physically, perceptually or conceptually. The user's conceptual model of a system is the knowledge that organizes how the system works and how it can be used to accomplish tasks. The conceptual model forms an integral part of the user interface.

3.2 *The need for human factors considerations*

The aim of human factors is to satisfy the largest possible proportion of potential users rather than to tailor the system to one user, particularly one with a detailed and sophisticated knowledge of the system. Therefore a proper man-machine interface takes account of the user's needs as well as system requirements. Poor quality will show up as a high proportion of input errors, loss of user confidence and motivation and high training costs. A high quality man-machine interface is based on a truly representative user model.

Recognized human factors literature has been used in the formulation of Recommendations Z.322 and Z.323. Where appropriate, human factor aspects have been incorporated into the texts.

Recommendation Z.322

CAPABILITIES OF VISUAL DISPLAY TERMINALS

1 Introduction

This Recommendation describes some of the capabilities which are important to the user and which are commonly available on interfaces based on VDTs. It is not an exhaustive list of capabilities. The use of additional capabilities, not covered in these Recommendations, is not precluded. Not all of the capabilities described need be present on a given system. Graphics capabilities are for future study and are therefore not considered in detail in these Recommendations.

System implementation of these capabilities may vary, depending for example on the degree of intelligence in the terminal itself and the allocation of responsibility for the man-machine interface among system components.

Items covered are treated from the point of view of the importance of their characteristics for designing the man-machine interface. Human factors are dealt with individually for each item.

2 Screen

2.1 *Character definition*

For further study.

2.2 *Cursor*

The cursor is important in the operation of a VDT because it directs the user's attention to that position on the screen appropriate to the task at hand, e.g., where the next character will appear. The cursor also allows the user to specify the location on the screen where an entry or change is desired by the user.

A set of general cursor qualities include:

— easily found by the user at any position in the display;

— easily tracked as it is moved through the display;

— does not interfere with the reading of the symbol that it marks;

— should not be so distracting as to impair the search for unrelated information displayed elsewhere on the screen;

— should be of a form that is unique and reserved for that purpose only;

— should be stable in respect to the position to which it is addressed until it is readdressed elsewhere as a result of user or system action.

2.3 *Screen partitioning definition*

The following definitions describe the physical partitioning of the VDT screen.

2.3.1 *Visible display*

The visible display is the entire physical screen of a VDT (see Figure 1/Z.322).

2.3.2 *Border area*

The border area is that part of a visible display which is physically unavailable for displaying or entering data (see Figure 1/Z.322).

2.3.3 *Display area*

The display area is that part of a visible display which is available for displaying or entering data (see Figure 1/Z.322).

Figure 1/Z.322, p.

2.3.4 *Window and window area*

The display area can contain one or more windows. A window contains a collection of related information. A window can consist of one window area or can be partitioned into window areas.

The different characteristics and operations specifying windows and window areas depend both on the system type and on the physical capabilities of the terminal.

2.3.4.1 *Window definition*

A window is a collection of one or more window areas which can occupy a part of the display area (sometimes the entire display area) and it is used for entering and/or displaying data. The collection depends on the application. A window is dedicated to an application. More than one window per application can be present at the samne time on the display area.

2.3.4.2 *Window characteristics*

The main characteristics of a window are:

— its name: allowing it to be identified;

— its location: relation to the other windows in the display area. Windows are displayed independently of each other. Windows can appear superimposed — one on top of another — or located side by side. When a window is located on top, it can hide a window or windows that are below it;

— the list of window areas it can contain;

— its size: its size expressed as height and width can vary;

— its state: a window can be ‘‘interactive,’’ or ‘‘not interactive’’. Information entry can be performed only when the window is ‘‘interactive’’;

— its visibility: a window is visible when it appears totally or partially on the screen. It can be partially visible either because it is overlapped by another window or because a part of the window is outside the display area;

— its limits: when it is visible, the limits of a visible part of a window must be obvious to the user;

— the application it is dedicated to.

2.3.4.3 *Window area definition*

A window area is a named part of a window that is dedicated for a specific purpose depending upon the application.

2.3.4.4 *Window area characteristics*

The main characteristics of a window area are:

— its name: allowing it to be identified;

— the purpose related to it;

— its presence state: a window area can be ‘‘present’’, or ‘‘not present’’. If a window area is ‘‘not present’’, it can not be seen on the screen whatever the position of the window it belongs to;

— its position in the window: the relative location of the window areas in a window should be fixed. This location can only be modified by changing the presence state of other window area(s);

— its size: its size expressed as height and width may vary;

— its visibility: when a window area is present, it can appear or not appear on the screen depending whether the part of the window it belongs to is visible or not;

— its limits: when it is visible, the limits of a window area must be obvious to the user;

— its text management: scrolling can be available in a window area.

2.3.4.5 *General rules for the display of windows and window areas*

A window can appear anywhere on the screen totally or partially in a non-restrictive manner.

Windows and window areas need not be displayed in all systems or in all applications or all the time in a given system.

The limits of windows and window areas must be unambiguously clear to the user. The techniques that may be used to achieve this include, but are not limited to the following:

— lines and boxes;

— inverse video;

— background colour. This use of colour should be distinguished from the use of colour as a highlighting technique where colour should be used in combination with other techniques.

Some examples of screens illustrating the use of windows and window areas are given in Figures 2/Z.322 to 5/Z.322. In these figures, windows are outlined by double-line boundaries while the boundaries between window areas are depicted by single lines. Lines and boxes are used simply as a concrete example that can be produced easily in print.

2.3.5 *Field*

2.3.5.1 *Field definition*

A field is a part of a window (sometimes the entire window area), which is used for entering or displaying information.

2.3.5.2 *Field characteristics*

The most important characteristics, which may vary in time, are:

a) its position within the window area;

b) its size;

c) its type:

— for entering information (input field): accessible for writing by user and system (e.g., default value);

— for displaying information (output field): inaccessible for writing by user.

The limits of an input field must be obvious to the user. There may be one or several fields within one window area (see Figure 6/Z.322).

FIGURES 2 à 6/Z.322, p.2 à 6

2.4 *Physical characteristics*

For further study.

2.5 *Video attributes*

Video attributes are used to emphasize certain important information, e.g., a title, a message, a chosen item, in order to attract the attention of the user. Video attributes work on the characters of the information shown within an entire window or window area, a part of a window or window area, an entire field or within just a part of the field.

The following video attributes may be provided singularly or in combination:

2.5.1 *Luminance*

For further study.

Information can be displayed in different levels of luminance.

2.5.2 *Colour*

Information can be displayed in different colours.

2.5.3 *Flashing*

Information can be displayed alternately as normal characters and as spaces in the prevailing background colour.

2.5.4 *Underline*

Information can be displayed with underlined characters.

2.5.5 *Size*

Information can be displayed in different character sizes.

2.5.6 *Font*

Information can be displayed in different fonts, e.g., italics, bold.

2.5.7 *Inverse video*

Information can be displayed by inverting the image of the characters, such as going from light characters on a dark background to dark characters on a light background.

2.5.8 *Concealment*

Information can be displayed as space characters, e.g., secret parts of a password.

3 Other output devices

For further study.

4 Keyboard characteristics

For further study.

5 Other input devices

For further study.

6 Transmission characteristics

There are two fundamental transmission mechanisms commonly employed and referred to as ‘‘character mode’’ and ‘‘block mode’’.

If a terminal uses character mode transmission, each and every character input at the keyboard is sent to the controlling processor one at a time. Thus, as is the case with the syntax of Recommendation Z.315, if certain regular keys have special meanings ascribed to them e.g., ; or !, then they can act as specific triggers to the controlling software which then performs some process on the preceding information in accordance with the given syntax rules.

If the same terminal uses block mode transmission, all of the regular typewriter keys and some of the special purpose keys only have an effect local to the terminal, i.e. the information input goes into the ‘‘memory’’ of the terminal and onto the screen normally, but not to the controlling processor. The implication is, obviously, that special actions assigned to these keys do not get processed until an explicit ‘‘send’’ action is made. A ‘‘send’’ action by the user is only required when information is to be moved from the terminal to the host processor.

The important point for the purposes of these Recommendations is that the use of a ‘‘send’’ key is not explicitly shown at any time. It is recommended that systems that employ ‘‘block mode’’ transmission either convey very explicit instruction on when a ‘‘send’’ action is required of the user or are designed to be able to accept and respond intelligently to incomplete input, i.e. ‘‘send’’ can be used by the user at any point without a fundamental disruption of the dialogue. As far as possible this will shield the user from the effects of the transmission mode employed.

7 Control functions

Control functions are those functions related to the man-machine interface that are applied by the user independently while in a dialogue with the system functions. Control functions have no direct impact on the system functions. Control functions are subdivided into cursor control functions and interface control functions.

7.1 *Cursor control functions*

A cursor is generally used as an indicator of the position where an action will take place, such as a character being written on the screen — either by the system or by the user. Cursor control functions do not directly affect the overall system state, but assist users in selecting data entry fields, editing fields, etc.

Examples include:

a) *Home position of the cursor*

Here ‘‘home’’ means a position in the display area to which the cursor can be consistently moved, from any position, by a single keystroke. The actual position in the display area which represents ‘‘home’’ may vary according to the activity being performed and the current layout of the display area.

b) *Movement control of the cursor*

Assuming that the VDT used supports direct cursor addressing, the following types of cursor movement are possible:

i) by the system, and

ii) by the user via cursor control functions. General cursor control functions independent of dialogue are:

— one line up;

— one line down;

— one place left;

— one place right.

Ideally, cursor movement should be easy to accomplish by means of a single, dedicated key for each function. Shifted characters should be avoided. If a cursor positioning control key is used, it should repeat when held down. Cursor movement may also be controlled by other input devices, e.g., light pen, trackball, mouse or joystick.

When cursor positioning is incremental by discrete steps, the step size of cursor movement should be consistent in both right and left directions and both up and down directions. However, the cursor may bypass inaccessible fields.

When character size is variable on the display, incremental cursor positioning should have a variable step size corresponding to the currently selected character size.

7.2 *Interface control functions*

Functions of this class are used to force specific actions relating to the interface. They are invoked by various means, including pressing dedicated control keys.

Examples of man-machine interface control functions include, but are not limited to:

— send (other words for the same function are ‘‘transmit’’ and ’’enter’’) [see § 6];

— editing control functions (insert character, insert line, replace character, etc.);

— capitals lock (the condition where letters are input as capitals only);

— select different font [see § 2.5.6];

— select different character size [see § 2.5.5].

Recommendation Z.323

MAN-MACHINE INTERACTION

1 Introduction

This Recommendation describes *how* | nteractions should take place between the user and the system from a logical viewpoint. It describes how an effective man-machine interface should appear to the user when utilizing the capabilities of VDTs as described in Recommendation Z.322. This Recommendation supersedes Recommendations Z.311-Z.317 for interfaces based on VDTs, referencing parts of them where appropriate. Specific human factor guidelines are included within the appropriate divisions of the text.

The capabilities of VDTs, e.g., multiple windows, inverse video, etc., when used consistently can lead to a more effective man-machine interface. Additional dialogue procedures are possible and often preferable with VDTs,

e.g., using different windows for different applications. Likewise, the transient nature of information presented on a screen may affect the selection of information display and the manner of presentation. The terminal capabilities available must be considered in conjunction with guidelines presented in this Recommendation in order to produce the most effective interface.

Many advances in the state of the art of man-machine interface design are incorporated in Recommendation Z.323. However, the use of graphics capabilities have not yet been considered in any detail in these Recommendations and must be studied further. The needs of the user moving between different systems or different types of terminals are best facilitated by ensuring that capabilities are used consistently and that user assistance is an integral part of interface design. Interfaces designed according to the principles outlined in this Recommendation will tend to be more user friendly, effective interfaces.

2 Common aspects

2.1 *Data display*

Data display is the presentation of information by the system to the user. During a dialogue the number, dimension and position of windows, window areas and fields in the display area may be changed. Not all fields, window areas or windows need necessarily display information at any one time.

Visual display terminals facilitate information entry through menu selection and form filling. Since presentation of more information at one time might cause confusion, care must be taken to label information clearly, to keep displays simple, to highlight information consistently and in moderation, and to maintain a consistent information layout as far as possible.

2.1.1 *General guidelines*

The layout of output is dependent on what type of data is presented. There are three basic types, combinations of which are possible:

— textual data;

— numeric data;

— tabular data.

a) *Guidelines for textual data* :

— text should be written using upper and lower case letters;

— abbreviations should not be used if confusion might be caused;

— plain text should be used rather than codes.

b) *Guidelines for numeric data* :

— strings of more than five numeric characters may be presented in groups of two to four;

— standardized formats (e.g., data and time as specified in Recommendation Z.316) should be used.

c) *Guidelines for tabular data* :

— in case of lengthy columns, spacing between about every five items improves readability;

— items which are related to each other should be placed close together;

— figures arranged in columns are easier to compare than figures arranged in a row;

— integers should be right justified;

— numerical entries with decimals should be justified with respect to a fixed decimal point position;

— text and labels should be left justified;

— if any text continues on another line, it should begin in the same column as the text above.

2.1.2 *Accessible and inaccessible parts of the display area*

VDTs provide the capability to characterize some fields of the screen as accessible for writing by the system only, some other fields accessible for the system and the user.

The fields used for the display of headers, parameter identities, delimiters, etc., should be accessible for writing only by the system (output fields). The fields used for the input of parameters should be accessible both to the system and to the user (input fields). The system can highlight these fields, for example, by underlining to distinguish the field or a default value, if appropriate. The user can access the field to input the desired value(s), to edit the previous input value(s), or to edit the offered default value.

The user may attempt to write into a field reserved for the system. This should not be allowed, an indication should be sent to the user and the input characters should be ignored. The type of this indication depends on the terminal facilities and may be an audible or visible signal. However, the terminal shall immediately recover from this situation so that the user can proceed.

2.1.3 *Highlighting*

Highlighting is used to emphasize visually a portion of a display area to make it stand out from adjacent portions, i.e., to call the viewer's attention to it. It should be used consistently and in moderation. In particular, care should be taken not to confuse or otherwise overload the user by highlighting.

There are a number of areas where highlighting may be applied, such as:

— defaults in forms;

— optional information entry in forms;

— indication of system irregularities and their urgency, etc.

There are a number of possible highlighting techniques, such as:

— different levels of luminance;

— colour;

— flashing;

— underlining;

— different character sizes or fonts;

— small or capital letters (lower or upper case);

— pointing with arrows, asterisk, etc.;

— inverse video;

— combinations of the above.

Some guidelines that should be followed in all applications of highlighting are:

a) when colour screens are used:

— in order to reduce problems for colour-blind users and to facilitate a transition between colour and monochromatic terminals within the same system, colour should normally be used in combination with some other means of distinction. Note also that some colours may have psychological associations, perhaps depending on the cultural tradition of a nation, e.g., red for danger, green for proceed;

— be consistent in the use of colour. Colour is a means to recognize rapidly particular windows, window areas or fields on the screen, independent of any system;

— colour should be used for additional distinction and emphasis. For example, colour should be used for aiding the user in locating information and for alerting the user to status changes. Colour should be used sparingly. It should not be used for purely aesthetic and nonfunctional effect as the main aim;

— if the user is given the capability to modify the colour of any area or object displayed on the screen, the user should be cautioned about changing colour via any assistance mechanism provided to the user. For example, in the case where the user is changing adjacent areas/objects to the same colour, a warning should be given. Where the capability is provided, the user should be allowed to make any modifications desired. Also, it is desirable that secure access to this capability be provided;

— the number of colours with specific meanings should be limited. Associating meanings with too many colours may confuse the user;

— colour combinations should be chosen such that there is sufficient contrast in hue and density wherever two colours meet. This is particularly true in the case where a text is displayed over a colour background;

— colour combinations should be chosen with care, as many combinations can be displeasing to the eye;

b) use only one level of luminance in addition to the normal level when highlighting. Variations in room lighting, specific VDTs and user perceptions make it unlikely that more than two levels will be universally distinguished;

c) when using more than one highlighting technique, do not highlight more than 30% of the display. If everything is highlighted, even differently, then nothing is highlighted;

d) since flashing attracts much attention, its use should be restricted to special applications, e.g., alarms. Once the user acknowledges the perception of the flashing information, the flashing should be stopped;

e) if the user needs to read text from a flashing area, the flashing should be slow in order to make the text readable. An alternative would be flashing pointers, pointing to the text area of importance;

f) in one system, or at least in each job area, highlighting facilities should be consistently applied;

g) information can be displayed with underlined characters. However, this type of video attribute might make it difficult to observe the cursor on terminals where the underline character is used as the cursor.

2.1.4 *Information layout*

A user should always be able to recognize at first sight:

— where parameter input is desired in a form;

— where system response is expected;

— where the system status is displayed;

— where user guidance is expected, if requested;

— where menus are displayed.

Therefore, the information layout, when determined by the system, should follow common rules in such a way that information of certain categories will be displayed in certain portions of the display area.

The information layout should be consistent in any one system. Information, which is not necessary in certain job areas, may be omitted.

2.1.5 *Description of window areas*

The following window areas can be distinguished in a window on the display area:

— *General information window area.* | This window area can contain system identification and/or application identification, and optionally, date, time, and other relevant information. This window area is optional;

— *Status window area.* | This window area should contain alarm indicators of the system being controlled, trouble reporting information from connected equipment, and message waiting indicators. The information displayed may be restricted to the particular application being controlled. This window area is optional;

— *Work window area.* | This window area should be used for information entry through form filling and menu-item selection. The work window area may also be used as a graphic display and screen editor area, and should support scrolling. This window area is required for information entry through form filling and menu-item selection but is optional otherwise;

— *Output and input window areas.* | These two window areas should support scrolling and should be user controllable in size. The input window area should be used for direct information entry. Response to the direct information entry as well as output outside dialogue should appear in the output window area. Input acknowledgements may also appear directly following the command in the input window area. The scrolling should occur in two window areas separately, or both window areas may be combined into one window area. These window areas are required for direct information entry but are optional otherwise;

— *Special keys and directives information window area.* | This window area should display function key labels and specifics about the use of directives. This window area is optional.

2.1.6 *Ordering of window areas*

The relative locations of the status, work, output, and input window areas should be fixed for any given system.

Screen layout recomendations for window areas that span the entire width of the window are shown in Figure 1/Z.323. In this case, the screen layout will have the window areas ordered as shown with the understanding that each window area remains optional.

Figure 1/Z.323, p.

2.2 *Input editing*

Editing mechanisms can be used to correct erroneous input during data entry or to change previously entered input in order to resubmit it.

Several possibilities of editing can be distinguished, including the following:

— delete last character or last n characters;

— delete or overwrite last field;

— delete or overwrite arbitrary fields;

— insert characters.

Editing mechanisms may be dependent on the facilities of a terminal, such as function keys.

2.3 *Response time*

In a system operating normally, response output (see Recommendation Z.317) to a command should be presented to the user within a psychologically acceptable time limit, normally taken to be of the order of two seconds after input. For any given type of command, this time limit should be as uniform as possible in order to meet the expectations of the user.

Depending on the nature of the command, two types of response output can be distinguished:

a) that which conveys the results of the execution of the command;

b) that which concerns the acceptance only of the command, results being communicated to the user by output outside dialogue.

Response output concerning user errors should be given to the user as soon as possible. Although a fixed rule cannot be defined, the following guidelines can be given:

— syntactical errors must be discovered very early by the system; the response time should be within the psychologically acceptable time limit;

— semantic errors can sometimes be discovered early, sometimes late, depending on the type of command and on the nature of the error; normally the feedback should be given to the user as soon as the error is detected;

— semantic errors in pre-scheduled jobs should be indicated to the user either immediately after the command input, if this is possible, or at the time the result is expected.

2.4 *Directives*

The presentation of system output in the form of guidance output, menus, form output, waiting system reports, next page, etc., can be controlled by means of input statements called directives. It is possible to qualify the effect of directives either by the use of context or by the use of additional parameters.

Directives are used to direct the system to present information rather than to execute a command; they can also be used in the interaction between the user and the system prior to command execution.

Directives can be given to the system by a word, e.g., HELP, by a special character, e.g., ‘‘?’’ (question mark), a dedicated function key, or by non-keyboard devices.

Directives can never cause any change in the state of the system. This distinction from commands is made to encourage users to make full use of such facilities without fear of altering the system unintentionally.

The subject of directives needs further study.

2.5 *User guidance*

When a user interacts with a system, there are times when more information about the system is needed than provided by the dialogue element in use, to assist the user in proper and efficient system operation. This information can be provided by means of various categories of user guidance.

Examples of different types of information that could be obtained in a guidance output are:

— how to obtain more specific guidance. A single guidance output at the highest level of simplicity might be displayed when the user enters a directive without any parameters, and the precise nature of guidance required is unclear from the context;

— general principles of dialogue procedure;

— what telecommunications services are available;

— what jobs can be performed;

— a description of structure and application of either classes of commands or a single command in detail. The user must specifically request that such output be displayed, either from the highest level of guidance output or via the parameter on the guidance directive;

— how the job is performed without actually executing it;

— what the user has done so far;

— what kind of entry the system expects from the user, e.g., possible commands, range of a parameter value, example of a correct parameter entry;

— the meaning and consequence of forms, commands, menu items, etc., which are displayed on the screen;

— the syntax or a short explanation of a specific command or job;

— a short description of a specific parameter, e.g., its default value or the permitted range of values.

In order to make the guidance as effective as possible, the following guidelines can be given:

— any guidance provided must be kept up-to-date and accurate;

— guidance should be available in a consistent manner throughout the system;

— unnecessary codes and abbreviations in guidance messages should be avoided.

A classification for user guidance based on user interface characteristics is presented in Figure 2/Z.323.

Figure 2/Z.323, p.

2.5.1 *Stand-alone guidance*

A stand-alone guidance facility can be used without necessarily accessing the function for which the guidance is provided.

2.5.1.1 *On-line training*

The primary purpose of on-line training is to supplement or replace other training methods such as classroom instruction, training manuals, or video courses. It can provide training on how to use the system (or parts of the system) for the first time, to refresh understanding, or to learn the system or function in more depth.

This type of information is provided as a separate function, and is designed to facilitate the learning or educational process.

The major difference between on-line training and other types of guidance is that training usually takes place in a ‘‘special’’ situation, intended to encourage learning. Because of the close relationship between on-line training and other guidance facilities, it is impossible to design or evaluate other guidance facilities without considering the training system.

Rudimentary guidance may be perfect for a trained user who occasionally needs a memory aid, while very elaborate on-line help may be needed for anyone with no previous training.

2.5.1.2 *On-line documentation*

The primary purpose of on-line documentation is to provide the user with a comprehensive body of information about a given subject related to the function. The major difference between on-line documentation and on-line training is that on-line documentation is meant to be used as a reference by users with a fundamental understanding of the function, hence is not a replacement for training. Although available as a stand-alone facility, on-line documentation may also be accessible during execution of the function. In this case, to avoid confusion with other types of guidance, the user should be notified, either implicitly by distinct format or explicitly by a message, that this help is also available as a stand-alone on-line documentation facility.

2.5.2 *Unsolicited guidance*

An unsolicited guidance facility provides user guidance when the system determines there is a necessity. Examples of unsolicited guidance are messages and prompts. Messages are issued to provide information on the current task, give status or completion messages for background tasks, or to notify the user of error situations. Prompts are issued as a result of an action request by the user. Messages and prompts are means through which the system provides feedback to the user, and assists the user in completing a dialogue with the system. They may request specific input, such as a request to the user to key required data, or to request the user to take a specific action, such as inserting a diskette.

2.5.3 *Solicited guidance (on-line help)*

Solicited guidance (also called on-line help) is a system’s capability to provide a user with information on how to use the system while using it.

This help facility requires the users to solicit the presentation of help information by means of an explicitly or implicitly stated request. The primary purpose of the on-line help facility is to provide a consistent and easy-to-use tool, that upon request, will give operational assistance necessary so that the user can make efficient use of the system to accomplish a work product.

Help text written using a consistent style is easier to understand and promotes user confidence. The following guidelines are given:

— sentences should be complete and concisely written. Detail should be limited to only what is needed for guidance on the requested item;

— sentences should be action oriented;

— help messages should use familiar wording so that users do not have to learn new wording for familiar concepts;

— references to outside material may be included in the help text, especially if the help information cannot be provided in a concise way.

2.5.3.1 *On-line help by implicit request*

This type of help facility assumes that the user, upon a specific interaction, requests information from the system. The basic distinction between unsolicited guidance and on-line help by implicit request is that the latter can be turned on or off by the user.

For example, the user employs information entry through form filling. If the on-line help by implicit request is activated, the movement of the cursor to a field reserved for the entry of a parameter value causes a message to appear in an output field reserved for implicitly requested help on form filling. The message describes the form in which the parameter value should be entered and the acceptable values. This approach has the advantage that the form layout does not need to be cluttered with supplementary information (as described in Recommendation Z.323, § 3.4.1).

In order to make this type of help facility effective, the following guidelines can be given:

— implicit requests should be limited to accompanying user actions that immediately proceed or are directly related to the entry of information (e.g., moving the cursor to an input field);

— the help displayed as a result of an implicit request should contain concise information that is of immediate use to the user;

— the help message needs to appear in a consistent location which is easily consulted but does not interfere with the information currently in progress;

— the implicitly requested help message should disappear automatically when the user moves on in the dialogue and the message is no longer relevant.

2.5.3.2 *On-line help by explicit request*

This type of on-line help (which will be referred to as ‘‘help’’ in this section for brevity) facility assists the user to complete a work activity by providing specific directions when explicitly requested by the user. The user indicates the item in question, and the system responds with the information specific to the request. Help output is displayed at the user’s request by the use of directives.

For systems providing this capability, the following guidelines can be given:

a) *Guidelines on information content and consistency*

— The information in on-line help should be designed to provide opertional assistance rather than covering training materials, or providing a tutorial;

— the help should be available within the context of the current dialogue. Contextual help means that within the appropriate authority level, the user can have assistance for items such as menus, options, parameters, commands, objects, or actions relative to the currently displayed information within current task of operation;

— the type and level of detail of help information provided should be consistent with the anticipated needs of the user at any particular stage of a dialogue. For instance, a ‘‘help’’ request made prior to inputting anything at a terminal could result in a high level introduction to the human-machine interface facilities, whereas a ‘‘help’’ request made instead of inputting a parameter value could result in detailed information on what possible values that parameter could have, and perhaps what each value means;

— the help facility should be designed to assist the user in progressing from one step within the dialogue to the next by supplying information that gives specific directions the user should follow;

— the help facility should be available throughout the conduct of any dialogue. For example, if help is available for one menu, appropriate help should be available for all menus;

— if the user requests help for an item that is not defined within the help facility, the user should be notified that no help is available for the specific item requested and be directed to help relevant to the context;

— if the system cannot determine exactly what help information is requested, it will present safe information such as a menu of topics instead of guessing at what the user wants;

— the help facility should allow a user to obtain information about dialogue elements which do not belong to the current context;

— the help facility should itself have help available. For example, this ‘‘help for help’’ could allow the user to select additional help topics, present a list of possible help items, or provide a brief description of the help facility.

b) *Guidelines on user-help facility interaction*

To provide a simple and efficient interface with the help facility, the following guidelines are given:

— help messages should preferably not overwrite data, error information or user commands and vice versa. In cases where this is unavoidable, a simple mechanism should be provided to retrieve the original information;

— the user interface to the help facility should be consistent with the interface to the other tasks within the system. For example, help menus should be constructed like other menus, selection techniques should operate the same, presentation style should be consistent, and command procedures should function the same;

— when a hierarchy of help information is required, the paths through the hierarchy should be short and simple;

— it should be possible for a user to request directly the exact level of detail required without having to step through intermediate higher level information;

— when possible, the help information should be displayed so that it preserves the visual reference to the dialogue content. Help information is most useful and least disruptive when the user has visual reference to both at the same time;

— where multiple pages of help are available, it should be possible to have any page displayed without having to display intervening pages;

— in the case of a long help message the user must be provided with some means of scrolling back or forth through the displayed text;

— instructions for exiting the help facility should be available on the system;

— when the user explicitly exits the help, the user dialogue should be restored to its original position before the help was requested;

— the help information should remain displayed either until the user explicitly exits the help facility or until the user executes a dialogue step which eliminates the need for the help information.

2.6 *Defaults*

In some applications the normal and most frequently used input can be predicted by the system. Default values which can be considered critical in the sense that they may create situations dangerous to the system integrity should be avoided.

2.6.1 *Use of default values during data entry*

To make the user's work easier, input of the most frequently used parameter values may be prepared by the system. If this offer does not match the user's desire, the possibility to overwrite the default must be open.

An offered default can be accepted by the user, either by active selection such as pressing a dedicated function key or by passive selection, i.e., without taking specific action.

The overwriting or deletion of defaults can be done by editing mechanisms as described in § 2.2.

2.6.2 *Display of defaults during data entry*

The main reason for using defaults is to simplify the user's information entry to the system.

To achieve this, defaults should be offered by the system and may be highlighted as described in § 2.1.3, so that it is obvious to the user which data entry area he has filled himself and which has been filled by the system. The highlighting technique should be consistent in a system or at least in a certain job area.

2.7 *Input error handling*

2.7.1 *Input error information*

In the event of erroneous input, some form of input error information, normally in the form of request output (see Recommendation Z.317), must be presented to the user.

Ideally, input error information would contain:

— where the error was detected;

— what kind of error was made;

— how to recover from it or at least how to find a way to recover from it.

In some cases it may be difficult to supply the user with all this information.

In many cases the input error information may be self-contained, in other cases reference may be made to other sources of information.

The length and detail of the message should be proportional to the nature of the error; the user should not have to look at a long explanation for a simple error.

Coded messages and intimidating jargon such as ‘‘syntax error’’ should be avoided. Messages should be polite and should not patronize or insult the intelligence of the user.

When an error is detected and error information is displayed, the field containing the error may be highlighted.

2.7.2 *Location of error information*

Error information should always appear in a consistent manner on the screen. This should be common within one system or at least within one job area.

2.7.3 *Multiple errors*

Multiple independent errors in one data entry should, if possible, be reported together at one and the same time.

Incidences of conflicting combinations of parameters or parameter values should be treated by the error information as a single subject.

2.7.4 *Correction of errors*

Following detection of an error situation, the user should be provided with mechanisms to correct the erroneous input. Such mechanisms could include:

— the system placing the cursor on the erroneous field and requesting input;

— the user addressing the field, e.g., by name, number or lightpen, or cursor control keys or joystick to get to the field(s) which needs to be changed.

The erroneous information should remain on the screen until it is corrected.

3 Dialogue procedure

3.1 *General*

In the general description of the dialogue procedure, aspects of error correction and of help request are not included. These topics are treated in the detailed descriptions of the specific dialogue elements. For examples of dialogue procedures, see Annex A.

3.1.1 *Structure*

The dialogue procedure is depicted in Figure 3/Z.323.

Figure 3/Z.323, p.

The dialogue is divided into three main parts:

— prologue;

— body;

— epilogue.

For the procedure prologue and the procedure epilogue, refer to Recommendation Z.317. The procedure body is depicted in Figure 4/Z.323.

Figure 4/Z.323, p.

3.1.2 *Dialogue elements*

In the CCITT MML, three different dialogue elements can be distinguished with respect to the method of entering information into the system via a man-machine terminal:

— direct information entry;

— information entry through menu-item selection;

— information entry through form filling.

Information entry can be accomplished exclusively by one of the dialogue elements or — if a system supports more than one dialogue element — by a combination of elements, e.g.:

— menu-item selection and direct information entry;

— menu-item selection and form filling.

3.1.3 *Selection of dialogue elements*

Choosing the right dialogue element depends very much on the nature of the job to be performed and the experience of the user. Often there are many different job areas that the user could deal with during his session at the terminal and the best method, for an inexperienced user, when selecting a job area, and then a specific job in this area, may be to use menu selection(s).

The experienced user would probably prefer a more direct method to reach a specific job, but will also use menu-item selection(s) when performing jobs that are infrequently used. Therefore the availability of both dialogue elements is attractive.

For maintenance staff who gain access to a system via the public switched telephone network with a simple portable terminal, it may not be possible to use every dialogue element due to restrictions imposed by the terminal characteristics.

Directives may be used for selecting dialogue elements. They may be either abbreviated menu or form identities, or function keys. The abbreviated menu or form identities need to be uniquely distinguishable from command codes, e.g., an abbreviated form identity could consist of a command code terminated by a question mark.

If direct information entry is available besides other dialogue elements, then direct information entry should always be possible after output of a ready indication or a menu. This may or may not require the use of a directive.

It should be possible to enter an allowed command or destination identifier even if a displayed menu does not contain it.

3.1.4 *Start and end of information entry*

The system invites the start of information entry by the output of:

— a spontaneous menu (one that is automatically given) and/or

— a ready indication.

The spontaneous menu given may be different depending on the authority of the user or the terminal involved. Any menu can always be requested by the use of a directive.

Completion of information entry always results in an Input Acknowledgement as is shown in Figure 5/Z.323 or in an appropriate error treatment.

Figure 5/Z.323, p.

As in Recommendation Z.317, an Acceptance Output may be followed by an Interaction Request Output.

3.1.5 *End of input indication*

In all dialogue elements the user may need to mark the end of input in order to have the information interpreted by the system. This can be done by some special indicators (see Recommendation Z.314) which contain an implicit end of input indication or by special function keys, e.g., ‘‘send’’. If more than one dialogue element is provided in a system, the end of input indication should be consistently used within each dialogue element.

3.2 *Direct information entry*

Direct information entry can apply to any area of application of the CCITT MML.

The direct information entry, recommended for operation and maintenance, installation and acceptance testing of SPC systems, consists of two sub-elements:

— destination prologue;

— interactive operating sequence.

See Figure 6/Z.323.

For both sub-elements, refer to Recommendation Z.317.

Figure 6/Z.323, p.

3.2.1 *Information entry*

Direct information entry may comprise:

— destination identifier to enable the destination of the information entered subsequently to be changed;

— command code to identify the type of activity to be executed;

— parameter values necessary to allow execution of a requested action;

— manual response as a part of an entering procedure requiring hardware manipulation such as operating switches, replacing equipment, etc.

These aspects are specified in Recommendations Z.315 and Z.317.

3.2.2 *Execution of a command*

A request to execute a command will eventually lead to acceptance or rejection output, refer to Recommendation Z.317.

3.2.3 *User guidance*

Refer to § 2.5.

3.2.4 *Guidance output*

Guidance output is in general related to a command and contains information such as:

— the complete block of parameters to be input for a specific command;

— that part of the block of parameters that is still to be input;

— the parameter next to be input;

— the fact that the complete parameter block has been entered and a request to execute a command can be given.

3.2.5 *Error correction aspects*

Input error information can be contained in guidance output or in request output (refer to Recommendation Z.317 and to § 2.7).

3.3 *Information entry through menu-item selection*

The essential advantage of menu-item selection as a way of interaction is that it can remove memory load from the user. The items available are laid out for inspection, and the way in which each item may be selected is obvious.

The task of performing any transaction using a menu is thus reduced to:

— scanning the items;

— finding the required item (if already known by the user), or deciding which item to choose (if not already known by the user);

— selecting an item.

The use of menus is particularly appropriate for applications where there will be many casual users or where there may be frequent interruptions to work at the terminal, and for activities which are infrequently performed.

Menus may be used as a means to arrive at a command code, to select a new destination or to assemble and to execute a command with all its relevant parameters. The system outputs a list of items (the menu output), from which the user can select the appropriate item. In a menu selection procedure, a selection of items from subsequent menu outputs may be needed.

3.3.1 *Display of the menu output*

The menu output (see Figure 7/Z.323) may contain several types of information:

— menu identity;

— menu items;

— additional information.

Figure 7/Z.323, p.

The information can be displayed in fields and/or given by highlighting techniques.

The *menu identity* | s displayed in a field at the head of the menu. It identifies the menu, preferably in a concise meaningful manner to allow an easy recognition of the nature of the menu.

A *menu item* | s displayed in a field that contains a brief description of the item and an optional selection identity. By inputting such an identity, a choice can be made. The selection identity should be displayed at the left side of this field.

The *additional information* | s intended to present more information to the user in order to aid the selection of an item from the menu, e.g., the text ‘‘Enter choice’’.

The menu layout in the window should be consistent throughout all the menus in a given system. Only one menu should be presented at a time, always displayed in its entirety.

3.3.2 *Item selection*

Refer to Figures 8/Z.323 and 4/Z.323.

Figure 8/Z.323, p.

The selection of an item can be done in two basic ways:

a) inputting the selection identity;

b) pointing at the item by using techniques such as cursor positioning, lightpen, touch screen, function key, etc.

Selection of more than one item from one menu is not permitted.

When using a hierarchy of menus, it may be helpful for the user to be able to return to the previous menu.

When the user notifies the system that he has made his selection, the system confirms the input by a new menu, a form output, or an input acknowledgement.

3.3.3 *User guidance*

During selection the user can ask for help at any moment. Besides, for general help information the user may ask for specific help information by inputting a specific help request.

The system reacts to the user with a clarifying text (refer to § 2.5).

3.3.4 *Error correction aspects*

The system can ask the user to correct his selection if this is not valid. The response is given in the form of request output (refer to § 2.7).

3.4 *Information entry through form filling*

Form filling is a useful method of information entry when flexibility is needed, for example when optional as well as mandatory items of data are required for command execution or handling of data stored in the system.

3.4.1 *Information entry*

When this data entry procedure is to be used, the system first outputs a form (in accordance with Figure 4/Z.323) that requires user input. The form contains a list of parameters identified by parameter identities. The parameter input fields are either empty or contain default values (see Figure 9/Z.323). The form has to be filled in by entering the parameter values required followed by an ‘‘end of input indication’’. For handling data stored in the system, at least the key parameter values have to be input in order to identify the data record. For a read or a delete operation, this is sufficient. For an add or modify operation more parameter values are required. They may partly be obtained by a previous read operation. Completion of the form is indicated by an appropriate ‘‘end of input indication’’.

As many parameter values can be given as desired before an ‘‘end of input indication’’. Parameter value input fields may be skipped if the parameter is not relevant or the initial or existing value is appropriate. Clarifying text is output when a ‘‘help request’’ is input. When the data input by the form is not accepted by the system, a ‘‘request output’’ is given to indicate that completion or correction of the data in the form is required. A sucessful operation is followed by an ‘‘input acknowledgement’’.

The ‘‘end of input indication’’ can also be used to request a next page if the form covers more that one screen. It can also be used to request continuation with a new empty form of the same type after completion. Mechanisms to control this capability are left for further study.

Figure 9/Z.323, p.

3.4.2 *Form output*

The form output (see Figure 10/Z.323) may contain several types of information:

a) form identity;

b) per parameter:

— parameter identity,

— parameter value input field,

— supplementary information;

c) additional information.

Figure 10/Z.323, p.

The above information can be displayed in fields and/or given by highlighting techniques.

The *form identity* | s displayed in a field at the head of the form. It identifies the command, preferably in a concise meaningful manner, to allow easy recognition of the nature of the form, and an optional identity for command reference.

The *parameter identity* | is displayed in a field and contains the parameter label and an optional parameter position which could be used as a reference in a request output. The parameter label is a text string as defined in Recommendation Z.314. The parameter position should be displayed at the left side of this field.

The *parameter value input field* | is an accessible field. Initially this field is either empty and should be filled in by the user, or the system may display in this field the default value which can be overwritten by the user.

The *supplementary information* | provides an explanation to the user, if required, to aid input of the parameter value. It may give information such as:

— whether the parameter is optional;

— in which form the value should be entered, e.g., in alphanumeric form.

The *additional information* | presents general information to the user with respect to the whole form, e.g., guidance on how to submit the form to the system after finishing the input of parameter values.

The information applying to a particular parameter (parameter identity, parameter value and supplementary information) should clearly be associated with that parameter, i.e., co-located. The position of the fields in the form should be consistent over the form. In any one area of application, they should be consistent from one form to another.

If punctuation is used for delimiting fields, the punctuation from the appropriate direct information entry technique should be used.

3.4.3 *User guidance*

During inputting parameter values the user can ask for help at any moment. Besides general help information he can ask for specific help information by entering a specific help request. (Refer to § 2.5.)

3.4.4 *Error correction aspects*

A consistency check on the set of parameter values in the form should take place after completion. Acceptance or rejection is communicated by ‘‘input acknowledgement’’ or ‘‘request output’’, see Figure 9/Z.323. Validation of value ranges may take place per parameter value input so as to identify range errors as early as possible. Request output as a result of a per parameter check is not shown in Figure 9/Z.323. The cursor and/or highlighting can be used to indicate which value should be corrected. The user can correct the indicated parameter values by changing the values and when complete, re-entering the form contents to the system. (Refer to § 2.7.)

3.5 *Displayed form*

The displayed form can be used to show a form which has already been filled in. The displayed form can only be used for reading and the information cannot be changed by the user. It can appear as an input acknowledgement.

3.6 *Guidelines for the design of menus and forms*

3.6.1 *Scope*

This section deals with the human-machine interface that utilizes the advantage of the input and output facilities offered by menus and forms. By using these guidelines, designers will get a more standardized layout of the various menus and forms.

3.6.2 *General guidelines for menus and forms*

Individual menus and forms should have an identity. Figures 7/Z.323 and 10/Z.323.

Identities should be consistently positioned, preferably on top of the menu or form. (Recommendation Z.323, § 3.3.1 and § 3.4.2.)

Menu and form layout in the window should be consistent throughout all the menus and forms in a given system. (Recommendation Z.323, § 3.3.1 and § 3.4.2.)

Each menu or form should ideally appear in its entirety, so that the user is able to see all the items or parameters at once. If the entire menu or form is not displayed in the window area, then an indication must be given of where the user is in the menu or form.

3.6.3 *Guidelines for menus*

3.6.3.1 *Appearance and organization of menus*

A menu should give hierarchical groupings of logically related items.

A hierarchy of menus should have the least number of levels possible considering the last guideline of § 3.6.2.

Menu items should have a clear and concise description of the choices available. The selection identity should be displayed at the left side of this description.

To avoid errors, special care should be taken to organize and label items in hierarchical menus in such a way that the scope of each item, or the likely result of selecting it, is as clear as possible.

3.6.3.2 *Movement between hierarchical or multiple menus*

If it is possible to go directly to the desired menu by combining menu-item selection identities, then the system should prevent the bypass of mandatory steps.

It should be possible to go backward through the hierarchy, step by step without the necessity of entering the identity of the antecedent menu.

The option to return directly to the main (top) menu should generally be offered.

3.6.4 *Guidelines for forms*

3.6.4.1 *Appearance and organization of forms*

Parameters should be organized into logically related groups. In addition, it may be possible to organize these groups in a hierarchical manner.

Within the primary requirement for good readability, the length of the form should be minimized considering the last guideline of § 3.6.2.

Parameter identities should follow the general guidelines for textual data.

3.6.4.2 *Navigation between input fields in forms*

It should be possible to move the cursor between input fields by a single operation, such as a keystroke. This means that it should be possible to move the cursor to the next or preceeding field in a sequence, or in the case of a form that contains logically related groupings of input fields, it should be possible to jump forward and backwards between the groupings, perhaps skipping several fields.

3.6.4.3 *Presentation of error information about menus and forms*

When errors are made they must be reported to the user in a manner that is most informative, enabling the user to make the quickest recovery.

In some cases it is not advisable to report how to recover from the error, e.g., security reasons.

The location in the window for the error information should be consistent thoughout all the menus and forms in a given system and should clearly be associated with the menu item or the parameter concerned.

4 Monologue output

A monologue output is any output from the system which occurs outside a dialogue. This includes output outside dialogue as described in Recommendation Z.316, system status and alarm information, function key labelling, date and time, etc. Usually, each type of monologue output occurs in an appropriate window on the screen. The occurrence of a monologue output may be accompanied by an audio signal or highlighting in order to stimulate user action, e.g., on alarms. In general, it is not helpful to output information on a VDT which is not immediately useful to the user.

4.1 *Output outside dialogue*

Output outside dialogue is a spontaneous output indicating a certain event, e.g., an alarm situation, or an output in response to a previously entered command, e.g., traffic measurement result. Output outside dialogue should not normally disrupt a dialogue in progress. There are several possible means of achieving this, e.g., message waiting indicators.

4.2 *System information*

System information is information related to the status of the system and may contain items such as:

— system status indicators;

— alarm indicators;

— message waiting indicator.

4.3 *Function key labels*

Function key labels may be displayed in the display area to inform the user as to what functions may be accessed via programmable function keys. They may be displayed as characters or symbols and with various highlighting techniques. It should be obvious to which function key each function key label is associated.

Consistency should be applied when assigning labels to function keys so that frequently occurring labels appear in the same position in the display area.

5 Time-out control inside dialogue

Subsection 5 of Z.317 applies except for the second time-out in which case the timing begins after a spontaneous menu output or ready indication.

ANNEX A

(to Recommendation Z.323)

Examples of dialogue procedures

A.1 *General*

In § 3 of the body of this Recommendation (Dialogue procedure), a number of dialogue elements have been described and Figure 4/Z.323 showing how various inputs and outputs are related has been introduced.

The purpose of this annex is to clarify how the various elements interact. This is done by showing in a number of examples how the interaction between the user and the system appears to the user.

It is important to bear in mind that the examples are intended only to illustrate some of the possibilities described in the dialogue procedure in § 3 of the body of this Recommendation and that they are not to be considered as Recommendations.

In the examples only three types of window areas are shown. These are, from top to bottom: work window area, output window area, and input window area.

The relative position of window areas in the examples is shown in Figure A-1/Z.323. The relative sizes of the window areas in this Figure are not significant, nor are the lines used to delimit the windows. The actual method of best distinguishing windows from each other is terminal dependent.

Figure A-1/Z.323, p.

It should be noted that help requests and input error handling are not treated in the examples, i.e., it is presumed that all commands and directives are entered correctly. Each figure shows both the output from the system and the following input made by the user. The user input is written in italics in order to distinguish it from the system output.

Examples 1 though 5 show input of commands, and examples 6 though 8 illustrate data base input.

A.2 *Example 1*

H.T. [T1.323]

{	{ *rien*

Table [T1.323], p.

H.T. [T2.323]

{	{ Command executed

Table [T2.323], p.

Figure A-2/Z.323, p.

A.3 *Example 2*

H.T. [T3.323]

{	{ *rien*

Table [T3.323], p.

H.T. [T4.323]

{	{ *rien*

Table [T4.323], p.

H.T. [T5.323]

	{
{	
	{
	<

Table [T5.323], p.

Figure A-3/Z.323, p.

A.4 *Example 3*

H.T. [T6.323]

{	{
	rien

Table [T6.323], p.

H.T. [T7.323]

{	{ *rien*

Table [T7.323], p.

H.T. [T8.323]

{	{ *rien*

Table [T8.323], p.

H.T. [T9.323]

{	{ Command executed

Table [T9.323], p.

Figure A-4/Z.323, p.

A.5 *Example 4*

H.T. [T10.323]

{	{
	rien

Table [T10.323], p.

H.T. [T11.323]

{	{
	< 3

Table [T11.323], p.

H.T. [T12.323]

{	{
	rien

Table [T12.323], p.

H.T. [T13.323]

{	{ Command executed

Table [T13.323], p.

Figure A-5/Z.323, p.

A.6 *Example 5*

H.T. [T14.323]

{	{
	rien

Table [T14.323], p.

H.T. [T15.323]

{	{ rien

Table [T15.323], p.

H.T. [T16.323]

{	{ *rien*

Table [T16.323], p.

H.T. [T17.323]

{	{ Command executed

Table [T17.323], p.

Figure A-6/Z.323, p.

A.7 *Example 6*

H.T. [T18.323]

{	{ *rien*

Table [T18.323], p.

H.T. [T19.323]

{	{ *rien*

Table [T19.323], p.

H.T. [T20.323]

{	{ *rien*

Table [T20.323], p.

Figure A-7/Z.323, p.

A.8 *Example 7*

H.T. [T21.323]

{	{ *rien*

Table [T21.323], p.

H.T. [T22.323]

{	{ *rien*

Table [T22.323], p.

H.T. [T23.323]

{	{ *rien*

Table [T23.323], p.

H.T. [T24.323]

{	{ *rien*

Table [T24.323], p.

H.T. [T25.323]

{	{ Command executed

Table [T25.323], p.

Figure A-8/Z.323, p.

A.9 *Example 8*

H.T. [T26.323]

{	{ *rien*

Table [T26.323], p.

H.T. [T27.323]

{	{ *rien*

Table [T27.323], p.

H.T. [T28.323]

{	{ *rien*

Table [T28.323], p.

H.T. [T29.323]

{	{ Command executed

Table [T29.323], p.

Figure A-9/Z.323, p.

ANNEX B

(to Recommendation Z.323)

Examples of windows

B.1 *General*

In § 2.3.4 of the body of this Recommendation a description of windows and window areas are given. (See also Figures 2/Z.323 to 5/Z.323).

The purpose of this annex is to provide some examples of the use of windows and window areas.

It is important to bear in mind that the examples are intended to illustrate the use of windowing only, and that they are not to be considered as Recommendations.

In these examples windows are outlined by double-line boundaries and window areas are outlined with a single-line boundary. This method of depicting windows and window areas is chosen as an example that can be shown easily in print. Actual methods used to distinguish windows will be terminal dependent.

B.2 *Terminal supervision*

This window is related to an application supervising the terminal the user is using. It may contain information about the terminal, about terminal directives (e.g., ‘‘window state change’’ function key), about active connections between the terminal and applications, etc. The window contains two window areas:

— general information;

— output.

Figure B-1/Z.323, p.

B.3 *Identification*

This window is related to an application managing the terminals which are local to the site the terminal is linked to. This application performs access connections to terminals with different applications. The window contains three window areas:

— general information;

— work;

— output.

Figure B-2/Z.323, p.

In this example, at any given time, the work window area is dedicated to menu/form input.

B.4 *Dialogue*

This window is related to a site operation and maintenance application. It contains four window areas:

— general information;

— work;

— input;

— output.

Figure B-3/Z.323, p.

In this example, not all window areas are simultaneously visible. Work (menu/form) and input window areas are exclusive of each other. The user can replace one of these displayed window areas by the other one through the use of function keys.

B.5 *System status*

This window is used to display alarm indicators by an application managing exchange alarms. It contains two window areas:

— header;

— status.

Figure B-4/Z.323, p.