National Language Information and Design Guide

Volume 1

DESIGNING ENABLED PRODUCTS, RULES AND GUIDELINES

National Language Technical Centre

IBM

National Language Information and Design Guide

Volume 1

DESIGNING ENABLED PRODUCTS, RULES AND GUIDELINES

National Language Technical Centre

1st Edition (September 1987)

Preface

In today's worldwide marketplace, products must be designed to provide support for national languages. Volume 1 of the *National Language Information and Design Guide* contains a set of rules and guidelines that are considered necessary to enable a product to provide that support. Like all other design considerations, the decision to follow particular rules or guidelines will be made after balancing various requirements with design constraints. To achieve maximum benefit from these rules and guidelines, however, you should make every effort to incorporate all of them into the original product design.

This volume includes the following parts:

- "Part 1: Introduction" defines the terms used in this volume and explains the subsets.
- "Part 2: Base Subset" contains rules and guidelines with associated notes and relevant examples for the base (left-to-right) subset.
- "Part 3: Bidirectional Subset" contains rule and guideline statements with associated notes and relevant examples for the bidirectional subset.
- "Part 4: Double-Byte Characters Subset" contains rules and guidelines with associated notes and relevant examples for the double-byte character subset.
- The Appendixes contain:
  - A list of the rules and guidelines to give the developer quick access to a handy checklist for self-assessment
  - A list of countries grouped according to their national language implementation characteristics
  - A list of standards organizations, standards, and national laws.
- A glossary that defines acronyms and terminology.

The National Language Information and Design Guide

The other volumes of the *National Language Information and Design Guide* are:

- *Volume 2, Left-to-Right Languages and Double-Byte Character Set Languages,* SE09-8002, contains reference information that is useful in implementing national languages that are written left-to-right and are represented by both single-byte and double-byte coded character sets.
- *Volume 3, Arabic Script Languages,* SE09-8003, contains general information about Arabic script languages and some conventions on language usage.
- *Volume 4, Hebrew,* SE09-8004, contains general information about Hebrew and some conventions on language usage.

Common User Access (CUA)

The Common User Access (CUA), a basic element of Systems Application Architecture (SAA), ensures consistency in designing interactive user interfaces. To ensure that proper national language support can be provided to the interfaces developed under this architecture, use this volume in conjunction with the CUA documentation.

Contents

Figures

Part 1: Introduction

The introduction provides specific definitions for some terms that are used throughout this volume. These definitions ensure that a particular meaning is applied to the rules and guidelines. This introduction also describes how the rules and guidelines are grouped into subsets and how to use the subsets.

Chapter 1. Definition of Terms

The *National Language Information and Design Guide* precisely defines certain terms and concepts. These definitions provide a precise meaning for the material presented in this volume.

Enable/Implement/Retrofit

Three concepts are vital to an understanding of national language support: enable, implement, and retrofit. These concepts are referred to frequently throughout the *National Language Information and Design Guide.*

Enable To *enable* a product means to design it in such a way as to *facilitate* the inclusion of national language functions and to design a product in such a way as not to inhibit the inclusion and usability of national language functions in other products.

During the design stages of a product, the developer must keep in mind that the architecture and the design have to be flexible enough to allow national language functions to be included in the product (possibly at a later time) without causing rework of the design. Remember that enabling does not refer to including any specific language but to designing a product so that any desired language can be included without redesigning the base product.

Implement To *implement* a national language on a product means to add national language functions to the design of a product when the design has been *enabled* to accept those functions.

Implementation refers to specific languages and to specific additions to a product.

Retrofit To *retrofit* means to redesign and modify an *unenabled* product to add national language functions.

Like implement, retrofit refers to specific languages and to specific additions to a product, but because the product is unenabled, redesign of the base product is necessary to accommodate the language additions.

An Enabling Analogy

The best way to understand the distinctions between the terms *enable, implement* and *retrofit* is to use an analogy. If a language function can be represented by a switch, then *enabling* for this function (switch) would require that a hole be provided in the base frame design. *Implementation,* then, consists of mounting the switch. No change to the frame is necessary. If the hole is not in the frame and you must change the frame by drilling before you mount the switch, then you are *retrofitting* the frame. Another product, attached to the frame, may not be directly involved with this switch, but it must take into account how the frame is enabled. To be *enabled* itself, it must not prevent access to the mounting hole for the switch.

Quite clearly then, *enabling* means providing the means, or not getting in the way of adding a new language function; *implementing* means adding that new function without changing the base, and *retrofitting* means changing the base before adding the new function.

Rules and Guidelines

This guide also makes a distinction between *rules* and *guidelines*. A rule describes an action that must be taken, or some condition that must be allowed for, to avoid a major impact on a product's ability to implement national languages. In other words, if the design of a product does not follow a particular rule, the consequences might be so severe as to prevent the implementation of an entire group of languages. The guidelines, on the other hand, are suggestions that, if followed, will ease the implementation. To disregard the guidelines does not affect the status of an enabled product. The national languages of the subset can still be implemented as opposed to retrofitted, but the ease of the implementation might be reduced.

Rules	The developer *must* comply with the *rules* to produce an enabled product.
Guidelines	The developer *should* follow the *guidelines* to get maximum value out of the enabling process.

In other words, the *rules must* be adhered to so that nothing will *prevent* implementation of any language(s), and the *guidelines should* be followed to ease implementation.

Chapter 2. The Language Subsets

The complete set of rules and guidelines are designed to enable a product to implement the national languages for all the countries listed in Appendix B, "Country Groupings by Language Characteristics." If languages are divided on the basis of their implementation characteristics, the rules and guidelines fall into subsets that naturally support those characteristics. By using all the subsets of the rules and guidelines, a designer ensures that a product is enabled for the national languages of all the countries listed in Appendix B.

The following characteristics form the basis for creating subsets of the enabling rules and guidelines for written languages.

1. Left-to-right languages using single-byte code pages

2. Bidirectional languages using single-byte code pages

3. Languages using double-byte code pages.

The first subset applies to left-to-right single-byte languages and contains the rules and guidelines that are common to all languages. The first subset is considered a base for all products. The other subsets deal uniquely with the bidirectional and double-byte languages.

Enabling for All Languages

The design direction is to enable all products for all languages. To do so, products will have to follow all the rules in all three of the subsets. The intent of enabling is to design products that will be independent of the language to be implemented.

Part 2: Base Subset

This subset contains the rules and guidelines that are fundamental to enabling a product. It is the basis on which the other subsets are built. If the base rules and guidelines are followed, a product will be enabled for the implementation of all languages that are normally written left-to-right, use both Latin-based and non-Latin-based alphabets, and are represented by single-byte coded character sets.

Chapter 3. Character Sets and Code Pages

Support for the national language of any country requires a minimum set of characters. In addition to this minimum set, other characters are needed to support application requirements. When all these characters are assigned code points, they form the basis of a code page. IBM, the International Standards Organization (ISO), and the Consultative Committees for International Telephone and Telegraph (CCITT) all maintain sets of standards that define these characters, character sets, and code pages.

Character Set Content

Character sets can be broken down into alphabetic, extended alphabetic, numeric, and special characters. For the most part, the alphabetic, extended alphabetic, and numeric characters are dictated by the requirements of the language and, in some cases, by the national laws. The special characters are in part dictated by the language punctuation requirements, in part by the accent requirements, and in part by common usage. Each language has its unique requirements and may not use the same characters in its character set in the same way another language does.

RULE 010:
The existence of a specific character set within a system or its components must not be assumed.

Enabling

A product must be designed so that *any* character set can be used. The system or device must be structured to be independent of any specific character set and must not assume easy access to a specific character either for software delimiters or hardware diagnostics. The only characters that are considered invariant throughout most EBCDIC character sets are contained in the so-called Syntactic Character Set and in the invariant part of ISO 646. Note that the invariant characters that are common to most EBCDIC character sets also are at the same relative position in most EBCDIC code pages. They have the same hexadecimal code point.[1] (The Japanese Katakana character set is a notable exception because it does not contain lowercase alphabetic characters.) A product that depends on the existence of a particular character outside the invariant set is considered not enabled, because it is not usable in all countries.

Alphabetic	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz
Numerics	1 2 3 4 5 6 7 8 9 0
Special Characters	. , : ; ? ( ) ' " / - _ & + % * = < >

Figure 1. Syntactic Graphic Character Set

Example

The left and right *bracket* symbols ([ ]) appear in many language character sets but do not exist in the character sets for France and Italy.

Example

File names automatically generated by a system module for use by system modules must not use special characters other than those considered to be invariant. If a control program generates a library with the name #LIBRARY, the character # will be displayed differently depending on the code page that is in use in the country. (File names owned by end users are not restricted to the syntactic set.)

Lowercase Characters

Some character sets, such as the Japanese Katakana, do not contain lowercase alphabetic characters.

GUIDELINE 1010:

Lowercase alphabets should not be assumed to be invariant.

[1] The special characters " and _ are not always in the same location in code pages, so they would have varying hexadecimal code values.

Selecting a Character Set

Along with the ability to work with any set of characters, provision must be made to select a particular set. Each country has one or more standard character sets that reflect the needs of the language(s) used in that country. When a product is sensitive to the content of a character set, there must be some means of selection.

RULE 020:
Character sets must be selectable on request by the operator, a user, or an application.

Enabling

How the selection of a character set is accomplished depends on the architecture of the product, the level of versatility that is desired, and the cost of manufacturing and maintaining the product. Selection through model number or country code is the historic method used on some products. Dynamic selection from a menu on a screen is also possible on some other more sophisticated products. A product must provide for some form of selection.

Example

Some keyboards are ordered from the distribution centers customized with a country-unique character set on the keys. The user selects the character set at the time of purchase. If the user wants to select another character set at a later date, the process can be as drastic as obtaining another keyboard or as simple as replacing the key caps. The control unit, however, must produce the appropriate code points for these characters. Selection of the character set at the control unit might be restricted to setup time, or there may be provision for dynamic selection.

The need for online or dynamic selection is best illustrated by looking at the multilingual environment. Multilingual ability is a growing need in both international communications and especially in internal communications of multilingual countries. Some countries require by law that communications to employees be in the language of the employee's choice, and, unfortunately, most standard character sets encompass only one language.

The character set for a particular language can be sufficiently different from that of another to make a simple extension of one language set to accommodate the other unworkable. In this case, a complete change of character sets is needed. In Switzerland, German, French, and Italian are all used within the same country. A French keyboard in a payroll department would lack sufficient characters to allow a correct entry of some German names. In this case, the capability to select a German character set in the middle of a session would be desirable.

Character Set Extension

The alternative to switching between character sets is to extend the character set being used. Because of special circumstances, a user may require frequent use of a unique character that is not part of the character set normally implemented on the keyboard being used but is part of the code page presently being used. Alternatively, there can be a character on the keyboard that is of little or no use for this application and can be eliminated.

Also there is a need for extension to characters not in the present code page.

GUIDELINE 1020:

Character sets should be definable by the operator, a user, or an application.

Example

A particular application can have a unique requirement to use the *registered trademark*, which is part of the resident code page but not part of the implemented character set, and, thus, does not appear on the keyboard. However, if no use is made of the *tilde* on the keyboard, the user should be able to redefine the character set supported by the keyboard to include the *registered trademark* and exclude the *tilde*. This extension to the character set will also have to be supported throughout the data path, both in input and on output.

Monocasing

Monocasing is the process by which alphabetic characters in one case are replaced by characters of another case.

RULE 030:
Monocasing must be definable for each language and code page.

Enabling

When the monocasing function is provided, do not merge it into the microcode or program code using a specific language or code page. Do not assume a monocasing algorithm is universally applicable. The capability to change the algorithm based on the language and country where it is used must be provided.

Example

In English, the lowercase *i* can be monocased to the uppercase *I*. Notice that the lowercase *i* has a dot over it, while the uppercase *I* does not. In Turkish, there are two sets of the letter *I*: a pair with dots over them and another pair without dots. The lowercase *i* with dot can be monocased only to the uppercase İ with dot, and the lowercase ı without dot can be monocased only to the uppercase *I* without dot.

In some languages, such as Arabic and Thai, most characters have only one case, unlike English. In these languages, monocasing has no meaning. Monocasing can still apply, however, to the Latin characters that have been included in the character set.

Folding

Folding is the process in which printable or displayable characters are substituted for those that cannot be displayed or printed on a particular device.

RULE 040:
Folding must be definable for each language and code page.

Enabling

Folding requirements vary depending on character set, language, country, and particular equipment in use. When folding is provided, do not merge it into the microcode or program code using a code page or rules which presume a particular national language.

Example

A printer might have a character set limited to 96 characters as in the case of some wheel printers. If the file to be printed has a character set larger than that available on the print wheel, not all the characters will be successfully printed. Those that cannot be printed must be replaced by ones that can. A single character, usually a space or dash, is used to indicate that a nonprintable character existed in the document file.

Code Points

To represent information in computers, a single byte of information known as a code point is used. Be careful to distinguish between the control code points and the graphic code points when dealing with hardware. They are used for different purposes.

Graphic Code Points for Hardware

RULE 050:
Single-byte character set (SBCS) code points in the graphics ranges must be used only for graphic characters and must not be used for control purposes in hardware.

The graphics ranges are as follows:

(EBCDIC)	X'40' to X'FE'
(ISO)	2/0 to 7/14, 10/0 to 15/15
(IBM PC)	X'20' to X'7E', X'80' toX'FF'

Note: In the ISO environment, 10/0 and 15/15 are sometimes restricted to have the meanings of "space" and "delete" respectively, for reasons of compatibility with a 7-bit code environment.

Note: In the IBM PC environment, X'01' to X'1E' andX'7F' may be graphics or controls depending on context.

Enabling

Graphic codes are used to display or print graphic characters. Control codes, on the other hand, directly perform a function such as executing the carriage return or ringing a bell. Each code should be used for its designated purpose according to the corporate standards. Code page compatibility problems arise immediately if the preceding rule is violated.

Example

If a hardware device uses what appears to be an unused graphic code point to supplement its control codes, it can prevent the proper use of a graphic character that uses that code point on another code page. This use might result in excluding the implementation of one or several languages on systems using that device.

Control Code Points for Software

The limitations to the use of code points for software control are not within the software. The limitations are imposed by the hardware usage of these code points. A code point cannot be used at the same time for both graphic and control purposes. The alternative control code points must be selectable.

RULE 060:
The use of a graphic character for software control purposes must not preclude the use of the same character in the text of messages, menus, prompts, or input or output fields.

Enabling

One way of applying this rule is to have those graphic characters used for control purposes available for redefinition. In this way, if a string indicator, such as & in &NAME, is used in text also, the string indicator can be changed to another character, such as %, to avoid conflict with the text.

Example

In many cases, the special characters of a graphic character set are given special meanings or special functions in a particular application. Most common is their use as delimiters between variables and commands. Each country attaches its own significance to the special characters, and what might seem to be an ideal delimiter in one language is so widely used in another that data entry is inconvenient for the user.

The following shows the type of problem that can occur when this rule is ignored.

'C01 address does not exist'	in English
'L'adresse C01 n'existe pas'	in French

The apostrophe is used to indicate the beginning and end of the message. Unfortunately, the French translation uses the apostrophe inside the text, so that the product considers that message to be only one character long.

Example

The space character, which is often used as a delimiter for commands, must conform to this rule as well. One example is the command,

UPDATE *variable*,

which looks for the space to mark the beginning of *variable*. When translated into French it becomes

METTRE À JOUR *variable*,

which has two spaces already in the command. The alternatives include either redefining the delimiter as an equals sign,

METTRE À JOUR = *variable*,

or replacing the imbedded spaces in the command by a character selectable by the country:

METTRE _À _JOUR *variable*.

Validity Checking

Often validity checking is performed on input data. The data entered is checked against some criteria before it is passed on to the application for processing. Data not meeting the criteria is rejected and withheld from the application.

RULE 070:
The set of characters allowed for use in the entry of data must be definable by the system operator, a user, or an application.

Enabling

Because the set of characters used to enter data is different for each language, it is necessary to allow the set to be redefined as the need arises.

Example

If a field is defined as alpha and only the 52 characters A–Z and a–z are allowed for data entry, the extended alphabetics that are used by most non-English speaking countries will be rejected.

Graphic Symbols and Icons

Avoid graphic symbols and icons unless you are certain that the symbol is universal. A symbol that is not universal and has no national equivalent cannot be translated.

RULE 080:
Graphic symbols and icons must be universal or translatable.

Enabling

Graphics and icons, such as common keyboard symbols for insert, delete, new line, new field, and cursor, are not necessarily the same in all countries. Various standards organizations, such as ISO, can be consulted to determine which icons are standard or have national equivalents. Icons, however, can be ideal for a multilingual environment.

Unassigned Code Points

Undefined code points on a standard code page must not be used arbitrarily for extra graphic characters. Unused code points are usually reserved by the standards organizations for evolution of future standards.

RULE 090:
Code points not currently assigned in standard code pages must not be assigned to new characters.

Enabling

By using these undefined code points indiscriminately for their own unique purposes, developers can seriously interfere with the migration of the product to future national standards and can also create compatibility problems with existing machines.

Defining Special Characters

Special characters (including punctuation marks) and their use should not be program dependent. Rules governing their usage vary from language to language. It should be possible to use special characters whenever necessary in the translated text.

GUIDELINE 1030:

Special characters, including punctuation marks, should be definable and not program dependent.

Example

When translating questions in Spanish, the translator should be able to precede the sentence with the ¿ (inverted question mark). This special character (punctuation) should not interfere with the proper functioning of the program.

Underscoring

The conventions for underscoring change from country to country. For example, in some countries, punctuation is underscored, but in others it is not. An option should be provided so that the characters to be underscored can be selectable.

GUIDELINE 1040:

The characters to be underscored should be selectable.

Chapter 4. General Design Considerations

All products, including applications, control software, controllers and hardware, must be designed to allow implementation of national language support. The rules and guidelines in this chapter are fundamental design considerations and apply to all products.

Character Accessibility

On occasion, it is convenient for a user to access characters from the presently active code page that are not in the presently active character set. This was discussed earlier in "Selecting a Character Set" on page 11 and in "Character Set Extension" on page 12. All levels of products, from hardware to applications, must allow such access. If a particular product in the chain, such as a controller, is not concerned with a given character, the product must be capable of passing the character information unchanged to the next level.

RULE 100:
All characters on the active code page must be accessible.

Enabling

Products intervening between the end user and an application must be able to pass all graphic character code points to and from that application. There must not be any form of filtering that cannot be changed to meet national language requirements. The application itself must be ready to handle all the graphic code points on the active code page.

Example

Code pages that contain the character set Latin #1 encompass the characters of the languages of most of Western Europe and North America and are, therefore, multilingual code pages. Individual language character sets use only a subset of the total characters on these code pages, but products attached to controllers can quite often provide access to the rest of the characters. The controller must not limit the scope of operation of these products.

Controller Support for Multiple Code Pages

To support multilingual applications, the controller should concurrently support any code page on any attached device. Any attached device can select the code page appropriate to the application in use. For example, one operator could be using the German code page while another could be using a French code page on the same controller.

GUIDELINE 1050:

Controllers should support concurrent handling of devices requiring multiple or different code pages.

Isolation of Language Dependent Hardware, Microcode, and Software

Any part of a product that must be changed for each new language or new country support is considered language dependent.

RULE 110:
Language dependent parts of a product must be isolated from nonlanguage dependent parts for easy modification.

Enabling

Language dependent items must be separated from the rest of the product.

If the language and country dependent areas of a product are partitioned or isolated so that these areas and nonlanguage areas can be changed independently of each other, the product is enabled. Although there may or may not be an effect on the initial cost of language implementation for nonenabled products, the major cost of not enabling (not isolating) may come later, for maintenance.

Example

If a character generator is to be implemented in ROM, placing executable microcode in that same ROM would violate this rule. The patterns placed in a character generator tend to be stable and not prone to change. Microcode, on the other hand, is prone to change either through enhancements or fixes.

Modular Microcode and Software Design

A modular approach to microcode and software design assists the implementation of languages. The modules should be easy to modify, and functions should be easy to change. The modular approach is particularly important when it comes to bidirectional control, where many modules can be affected.

<u>GUIDELINE 1060:</u>

Microcode and software should be modular and designed to be easily modified.

Independence of National Language Support for Components

Design for national language support will often involve several components of a product, such as data stream handling, display handling, keyboard handling, and printer handling. Each of these components must be supported independently of each other.

RULE 120:
The design of a product must allow for the national language support of the various components of the product to be independent of each other.

Enabling

A product must be designed in such a way that changes or additions to the national language support of one of its components has no effect on the national language support of any other components of the product.

Example

It must be possible to add support for a new keyboard without affecting the display operations or the handling of the data stream.

National Language Exits

Quite often, the execution of mainline processes must be interrupted and changed to meet the requirements of the national language being implemented. It is important that exits are provided that allow special country or language dependent processing to be performed.

RULE 130:
National language exits must be provided at strategic points.

Enabling

Products that include the appropriate exits as part of the design are enabled. If a module of executable code in the base product must be rewritten to retrofit a new language, the product is not enabled.

When code is not enabled, development time is increased because the base module must be rewritten and retested as part of retrofitting a new language. Also, the maintenance of base functions means a parallel effort in the customized language version.

The scope of this rule is all of the functions mentioned in this document, including those of Chapter 9, "National Usage Considerations." When a product calls for Linguistic Functions (LFs), great care must be taken to ensure that adequate exits are provided to allow implementation of national language versions of those LFs. Adequate exits to allow implementation of the functions of the Bidirectional and the Double-Byte Character Subsets must also be provided.

Example

Some of the special processing that might require exiting to a language-unique routine could include generating special check digits, bidirectional controls, and data manipulation, such as, monocasing, folding, sorting, searching, and shape determination.

Diagnostics

Depending on the overall level of national language implementation, diagnostics may be translated. In several countries, there is a legal requirement for the service personnel to be able to work in their national language. In addition, for many products, the customers are expected to run the diagnostics before calling for service. All the rules and guidelines apply equally to diagnostics as they do to any other product.

RULE 140:
Diagnostics must be enabled.

Chapter 5. Electronic Character Generation for Displays and Printers

This chapter deals with national language support considerations in the design of displays and printers that use character generators.

Character Generator Size

Although many coded character sets consist of fewer characters, a character generator must be capable of creating a minimum of 190 characters when dealing with single-byte character set (SBCS) code pages. This standard is consistent with the new extended 8-bit ISO standards and the EBCDIC standards within the IBM Corporation. Standards for DBCS within IBM are extensions of EBCDIC standards and are discussed in "Part 4: Double-Byte Characters Subset" on page 83.

Special icons and indicator symbols are not considered part of the language character set. They are produced by the system or application to alert the user to a particular condition. Therefore, their images should be generated separately from any language character. They can, however, for convenience, reside in the display or printer's character generator.

RULE 150:
Character generators must produce at least 190 user-accessible graphic characters, excluding indicator characters and special icons.

Enabling

All 190 graphic characters on a code page must be displayable and printable even if only a subset of the character set is in current use.

Example

There are generally 190 graphic characters in a single-byte code page. If a character set with only 127 characters has been implemented on a keyboard, there are still 63 more characters on the code page that must be accessible by other means.

Note: The 190 characters described above do not include the code points X′FF′ and X′40′ (space) in EBCDIC and 7/15 and 2/0 (delete and space) in ISO, which are classed as nongraphic characters.

Character Cell Matrix

The size of a character generator is not only governed by the number of characters it contains, but also by the number of picture elements (pixels) that make up the individual character cell matrixes.

Some of the more complex characters required by SBCS languages are difficult to represent on a screen or printer if there are fewer than nine individually displayable pixels across the character cell horizontally.

Character cells should be high enough to allow for readable descenders for lowercase Latin and non-Latin characters. Underscores should not interfere with descenders. Character cells should be high enough to allow for readable accents on uppercase characters. In general, the accents on uppercase characters are poorly rendered today. Character cells 16 pixels high provide readable results for SBCS.

GUIDELINE 1070:

The matrix of pixels forming a character cell should be not less than 9 pixels wide and not less than 16 pixels high.

Each pixel should be individually accessible for the construction of the shapes required in a language and the hardware should not force inter-character spaces. (See also “Cursive or Touching Characters” on page 81). The lack of a pixel in a strategic place could change the meaning of the character and possibly the word in which it exists.

Example

Figure 2 illustrates how the vertical minimum of 16 pixels and the horizontal minimum of 9 pixels is needed for quality character shapes.

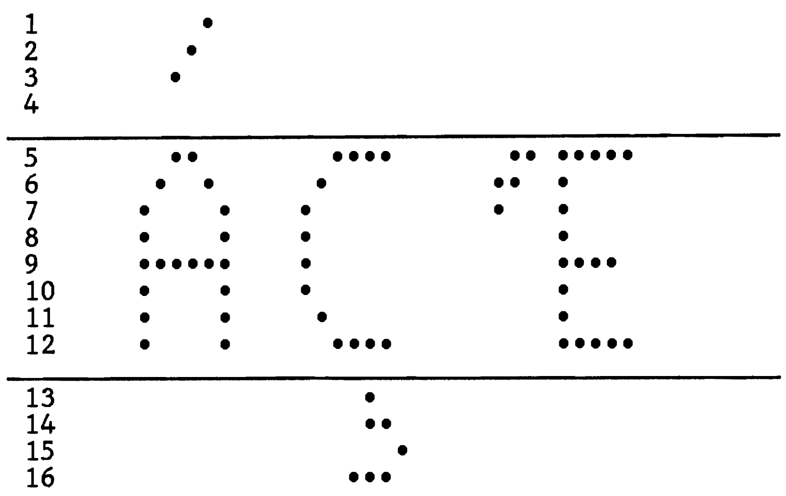

Figure 2. Character Cell Matrix Example

Flexibility of Character Generators

Once the character generator has been designed to enable it to contain 190 usable characters at one time, the next step is to make the content changeable. All languages cannot be represented with one set of 190 characters. The ability to implement other sets of 190 characters must be provided.

RULE 160:
Character generators must be built to be flexible so that the contents can be modified and expanded.

Enabling

Read-only-memory (ROM) character generators, soldered to the card or board, are not easy to modify. Replacing them for a new language generates new board or card part numbers. Using pluggable ROM, programmable ROM (PROM), or erasable programmable ROM (EPROM) is a better approach. From a language point of view, the most desirable type of character generator uses RAM, which gives maximum language flexibility.

Example

Most languages that use Latin-based alphabets such as English, French, German, and Italian can be supported by a common set of 190 characters. A very different set of characters must be provided, however, to support Greek, Cyrillic, and other non-Latin-based languages.

Capacity of Hard Font Printers

Along with the need to display at least 190 characters is the need to print the same character set conveniently.

GUIDELINE 1080:

For hard font printers, the capability of printing at least 190 characters without a print element change should be provided.

Having to change print elements part way through a document to pick up characters from another element is not convenient. A printer should be able to print a single-byte code page using only one print element. All print elements, however, do not have to contain all the characters on a code page. Subsets of the code page can be provided to allow for increased speed on machines such as belt printers. It is key, however, that the full code page can be printed from one element if the customer so desires.

Brightness of Horizontal and Vertical Lines

In many languages, complex characters make it desirable for character elements to be one pixel wide to fit the character into a cell of minimum dimension as described earlier. However, it is critical, for character readability, that character elements in both horizontal and vertical directions appear at the same brightness or density.

RULE 170:
The brightness or density of a character element must be consistent for both horizontal and vertical lines.

Enabling

When a character element is formed by more than one pixel, the brightness of the combined pixels in the character element must be equal. A product that requires combinations of pixels (double dotting) to provide equal brightness effectively reduces the usable pixel count in that direction and must compensate with additional pixels.

Example

If a product used a 4x4 pixel character cell matrix, it would seem perfectly capable of displaying a capital H in that cell. If, however, the vertical elements were less bright than the horizontal element, unless a double pixel width is used, the H would be impossible to display without uneven brightness in the elements. Although this is an extreme example, the equivalent situation can occur despite the implementation of a much larger character cell matrix.

Electronic Font Printers

A restriction similar to the one described for displays can occur on electronic font printers. Here, because of design constraints such as limited duty cycle, the printer prohibits the use of successive pixels. This restriction reduces the number of individual pixels that can be actually addressed for a particular character, resulting in a much reduced effective matrix size.

RULE 180:
For electronic font printers, a mode must be provided such that each successive pixel of a character can be printed.

Enabling

When the matrix size of the printer's character cell is the minimum, as stated earlier, there must be a way for all successive pixels to be addressed and used. Where this is not possible, the matrix must be increased in size to provide the appropriate effective minimum.

Example

In some languages, such as Thai and the Arabic-based languages, the quality resulting from using every other pixel is very poor, almost to the point of being unreadable.

Chapter 6. Keyboard and Keystroke Processing

Keyboard Concepts

The keyboard and its interface logic are used to input data and to activate functions. The national language features of a keyboard are character sets, data or text entry functions, and labeling of keys. In this chapter, the rules and guidelines for keyboards are explained using the following terms.

Keyboard Layout

A keyboard layout consists of a graphic layout and a function layout. In this chapter, only the graphic layout is discussed. A *graphic layout* is created when all the characters from a character set are arranged on a keyboard. Application and country requirements determine the character set that will be used to create a layout. While classes of applications in a given system and country may share a single layout, other applications may require a completely different character set and layout. For example, numerous data entry applications can share one layout, but APL requires a separate layout.

A layout and its character set and code page(s) are bound together by the keyboard processing logic and the application's input processing component.

Physical and Logical Layouts

Every layout is a logical association of characters to key positions and is therefore a *logical layout*. The layout that is most commonly used in a system is labeled (printed or engraved) permanently on the keys, and is called the *physical layout*. More than one logical layout can be represented on a physical layout; for example, the U.S.A. English and APL keyboards. The labeling is governed by usability considerations.

Graphic Shift Levels and Layers

The placement of the graphic characters on the keyboard is governed mainly by country conventions and standards. If the number of characters is small enough, they can be arranged among three *shift levels* (lower, upper, and alternate (ALT)). An arrangement of characters on the graphic keys using a single combination of these three shift levels is called a *graphic layer.*

If the character set is too large for a single layer, then additional layers may be used to contain the remaining characters. Some characters may be repeated in another layer for ease of use. Multiple layers are also required when country practices or standards dictate them. A layout, therefore, consists of the graphic layer or layers needed to hold a particular arrangement of a character set.

The Greek keyboard is a typical layout with two layers; it has the Greek alphabet on one layer and the Latin alphabet on another, with numerals available on both layers.

Another example of a multiple-layer layout is a keyboard supporting graphic applications using programmable symbol sets. Each set of symbols is arranged on its layer, and these layers combine with the national layer to form a complete layout.

Software Control of Keyboards

Applications, and software in general, have a need to control their environment. A key component of this environment is the keyboard. An application may need to select and control the keyboard operating mode to impose uppercase, for example, or restrict the use to a specific language or layer.

RULE 190:
Keyboard design must allow for software selection and control of its mode of operation.

Selectable Language Graphic Layers

In countries where Latin and non-Latin alphabets are used, such as in Greece, the active graphic layer should be settable to either Latin or non-Latin under application program control or by the keyboard operator.

Example

One method used for switching languages, that is, switching between the first and second layers of a two-layer, bilingual keyboard, is to use the ALT key with the left shift key to access the first layer and ALT key with the right shift key fo: the second layer. The shift keys operate normally when the ALT key is not depressed.

In addition, the application must also be capable of automatically switching between layers if characters of a particular language must be entered next.

Selectable Keyboard Layouts

Because the users may want to redefine the character set, as noted in "Selecting a Character Set" on page 11, they must be given a means of key entering the new character set. Keystroke processing logic must be designed to allow the key entering the new character set to happen.

RULE 200:
Logical layouts different from a given physical keyboard layout must be available to the user.

Enabling

A wide range of methods can be used to enable keystroke processors to respond to a logical keyboard layout as opposed to the actual physical layout. In some cases, the processor might not even know the physical layout and must rely entirely on the logical identifier entered by the user. Where the physical identifier is known by the processor, provision for the logical layout to supersede it must be made.

For multilingual capability, the selection process must happen during processing. The physical keyboard layout will be the one used most of the time. However, for multilingual document creation and file transfers or for accessing remote data bases, the user must be able to select an appropriate alternative keyboard layout. This "shadow" keyboard layout can be a user defined one or one that IBM supplied with the system. The selected keyboard layout becomes the active keyboard layout.

Example

To reply to an inquiry from a French-Canadian, an English-Canadian must be able to redefine the keyboard content (layout) to contain the character set of the French language. The physical layout will not change, but either through a template or some other means, the extra character locations can be illustrated.

Definable Keyboard Layouts

Sometimes the users do not want to select an alternative layout that has been supplied, but would rather define their own. They should be able to move the present content of the keys around, or replace some characters with others from the active code page. They should be allowed, with the appropriate support in the rest of the system, to assign a code point and a key to a graphic that is completely outside the active code page.

GUIDELINE 1090:

Logical keyboard layouts should be user definable.

As explained in the section "Character Set Extension" on page 12, there are requirements to extend character sets. Because keyboards and their controllers are primary input devices, they should have this flexibility.

Keyboard Size

There are standards governing keyboard graphic layouts. Some of these national keyboard standards are listed in Appendix C, "National Language Standards and Laws." The number of graphics on the keys is dependent on the application and the country, but most of the keyboards require 48 graphic keys.

RULE 210:
If a product uses data or text processing keyboards, at least one of them must have 48 or more graphic keys.

Enabling

A product can support any number of keyboards. Some of them may have fewer than 48 keys for graphic characters. One or more supported keyboards must have 48 or more graphic keys for the product to be enabled.

Shifting between Language Graphic Layers on Keyboards

Some multilayer keyboard layouts contain support for several languages. The layers are arranged so that each will support a specific language.

RULE 220:
The product design must allow the implementation of at least two graphic layers of three shift levels each.

Enabling

Sufficient capability must be provided in the keystroke translate routine to handle the three levels that are accessed by the upper, lower, and alternate shift levels and the first and second layer switching needed to support multiple alphabet languages. Although two layers are a minimum requirement, present architectures provide for two or more layers.

Example

Cyrillic and Greek character sets each include both non-Latin and Latin alphabets. These alphabets coexist in a single layout of a multilayer keyboard. Each layer is dedicated to a specific alphabet, and each layer contains three shift levels.

Shift Indicators

There should be provision for a meaningful indicator that tells the operator the graphic layer the keyboard has been switched into. In addition, there should be an alternate shift level indicator as well as the upper and lower shift level indicators presently employed.

GUIDELINE 1100:

Indicators should be provided to show the level and the layer a keyboard is operating in.

Example

Level indicators are usually provided for the upper and lower shifts of a keyboard. Some products use an arrow on the status line at the bottom of the screen. Other implementations use a light emitting diode (LED) on the keyboard itself. Layer indicators have appeared as mnemonics in the Operator Information Area (OIA).

Positioning Graphics on Keys

For keyboards that shift between multiple languages, positioning graphics on keys is a complex task. The relative position of these graphics is determined not only by country convention, but by the primary language, the direction the language is written in, and human factors.

RULE 230:
The manufacturing process must not restrict the relative positioning of graphics within the valid area on the top or front face of the key.

Enabling

The Latin characters a through z should not be preplaced on keys forcing all other characters to be positioned around them.

Example

On bilingual keyboards using Latin and non-Latin characters, the primary language characters, generally the non-Latin characters, are placed corresponding to the starting position of the writing line in that language.

Greek is written left to right; therefore, the Greek characters on a Greek/English keyboard are placed on the left side of the keytop. Similarly, Hebrew is written right to left, and the Hebrew characters are placed on the right side of a Hebrew/English keytop.

In some cases, a third level is used, and the characters for this level are placed on the front faces of the keys. The designers must be able to place these characters at any point within the valid area, and not be restricted to starting on one side only. Some languages require the character(s) to be right justified.

Capslock

Capslock has been implemented on several products as an alternative to shiftlock. By definition, capslock locks the keyboard into a mode that produces the uppercase or capitals of all alphabetic characters. Shiftlock locks the keyboard into the upper shift for all graphic keys, not just alphabetic keys. Where there is no uppercase equivalent or duality of case (Katakana), then the character remains unaffected.

RULE 240:
Capslock must monocase to uppercase.

Enabling

The capslock function must not be sensitive to the level or graphic layer in which the character, in its lowercase or uppercase form, resides. Uppercase characters must be unaffected by capslock.

Example

On some keyboard layouts, lowercase alphabetics appear on the lower, upper, and alternate shifts of the keys. When capslock is invoked, the uppercase equivalent for these characters will be entered into the data stream. The uppercase characters must result even if the Shift key is used to access the character in the upper shift.

Capslock With Shiftlock

When a keyboard layout consists of multiple language graphic layers, the requirement for capslock and shiftlock may change between layers.

RULE 250:
Keyboard design must be able to support capslock and/or shiftlock or neither on each layer.

Enabling

Keyboards that provide support for more than one language must be capable of providing lock functions appropriate to each graphic layer.

Example

There are four possible choices for each keyboard layer. A hypothetical keyboard with 4 layers might be laid out as follows:

layer 1:	no capslock or shiftlock.
layer 2:	capslock but no shiftlock.
layer 3:	shiftlock but no capslock.
layer 4:	both shiftlock and capslock.

Defining Nonescaping Keys and Repeat Keys

Accents can be implemented as "nonescaping" keys to extend the number of accented alphabetic characters that can be entered from a keyboard without adding more keys. When these keys are depressed, the cursor does not advance until an alphabetic character or a space has been entered.

The positions and quantities of nonescaping keys and repeat keys depend on the language. Sometimes, one shift of a key is *nonescaping* while another is *normal.* Unless the *repeat* function is common to all data keys in all shifts, the selection must be changeable by language.

RULE 260:
Any data key must be definable by the user to be nonescaping or repeating.

Enabling

Nonescaping key function and repeat key function must not be locked into specific physical locations of the keyboard. In implementing new languages, the user of the base code must have the flexibility to define which of the 48 data keys in any shift will exhibit these functions.

An extension to this rule takes into account the capability of end users to define keyboards and character sets. These users must be able to define which keys are to be *repeat* keys and which are to be *nonescaping* keys. They must also be able to define the valid nonescaping key combinations for any character set defined by the user.

Keyboard Identification

The identification of the physical layout on the keyboard provides a known starting point for the system and diagnostics. This identification is particularly important when many keyboards with different physical layouts are attached to a system.

GUIDELINE 1110:

A keyboard should be capable of identifying its physical layout to the keystroke processing logic.

Nomenclature

A standard set of nomenclature, for example "CURSR BLINK," "DUP," and "ATTN," and its translation into a specific language must be used to maintain consistency for the keyboard operator. Before a new function is added to the keyboard, its nomenclature must be established considering the function definition and existing nomenclature. The function definition must be clearly described to the translator so that the translation of the nomenclature can be meaningful.

Chapter 7. Machine Readable Information (MRI).

Machine readable information (MRI) includes all textual information that is presented to or received from an individual interacting with a system. This includes menus, prompts, messages, report headings, system commands, and responses. Because control information (for example, attributes for highlighting, protected fields, or presentation direction), although not MRI, can influence the text, it must follow MRI rules.

Isolation

The concept of isolation (or separation) is basic to national language support. MRI isolation is critical in enabling for efficient and accurate translation. There are three groups or levels of isolation:

- Source level
- Packaging level
- System resident level.

Isolation of Source Level MRI

The term *source level* describes the MRI text that may be translated. Associated executable code must not be translated. The translator works directly with source level MRI to produce the national language version.

RULE 270:
All MRI and presentation control information must be isolated from the executable code.

Enabling

Products (software, hardware, or microcode) that contain MRI must isolate the MRI from executable code. Isolation prevents the translator from inadvertently creating functional problems during the MRI translation process.

If isolated at the source level of modules, MRI can readily be translated independently of ongoing development of executable code. Because these modules must contain no executable code, stable modules can be sent for translation while the executable code accessing these modules is still being developed.

Note: Some MRI has had to contain executable code because it was required by older dialog managers. If it is minimal, consistent, and stable, this mixed MRI has less of an impact on translatability than other forms of executable code, but it is not necessarily a feature of dialog managers.

Example

MRI isolation implies having the MRI stored in a separate module, table, data base, file, or hardware component.

Isolation of MRI at the Packaging Level

Software systems are packaged and delivered using several types of media. Most common are diskettes, tapes, or disks. Each system has its own method.

Isolating MRI at the packaging level simplifies the build process for NLS versions. Non-MRI and MRI upgrades can be done more easily and independently of each other. Also, implementation of new languages is simplified because the non-MRI code does not have to be reinstalled.

GUIDELINE 1120:

MRI modules should be packaged separately from the executable code.

MRI modules are compiled and written to storage media separately from non-MRI modules so that the two types of code can be separately installed and maintained. It also permits the use of multiple languages with one version of executable code, which would not be possible if they were packaged together.

Example

Consider a diskette-oriented example. The non-MRI should be packaged on diskettes separate from MRI diskettes.

Isolation of MRI at the System Resident Level

Isolation at the system resident level allows all MRI modules to be in a library that contains only MRI. The non-MRI code would be in another library. This separation enables a software design to have several MRI libraries on the system at one time with only one non-MRI library containing all executable code. The system, then, could allow the system user to specify the MRI library to be used for any given session.

GUIDELINE 1130:

MRI modules for SBCS systems should be loaded separately from the executable code.

Note: This is a rule in the double-byte character subset.

Multilingual support must start with isolation of the MRI modules at the system resident level. Many other details, including those concerning hardware, must be designed into the system.

MRI Expansion

English phrases are usually shorter than equivalent phrases in other languages; therefore, the space occupied by the English MRI will usually be insufficient to contain the translated equivalent of that MRI. The additional space required to support translation will vary according to the length and composition of the English text. Space requirements will also vary from language to language for any given sentence.

RULE 280:
Sufficient space must be available for MRI expansion caused by translation.

Enabling

There is no firm rule that can guarantee a particular amount of expansion room will be sufficient. Generally, short phrases require a larger percentage of space for growth than is required for long phrases.

Figure 3 on page 42 shows the relative growth that can be expected in MRI, and can be used as a rule of thumb when estimating and allocating expansion space.

English length	Additional space
up to 10 chars.	101 - 200 %
11 - 20 chars.	81 - 100 %
21 - 30 chars.	61 - 80 %
31 - 50 chars.	41 - 60 %
51 - 70 chars.	31 - 40 %
over 70 chars.	30 %

Figure 3. MRI Expansion Factors

These expansion factors assume no abbreviations are used.

Notes:

1. *When calculating expansion space, you do not have to include areas such as numerals or fixed length fields that will not expand; however, spaces and punctuation marks should be included. First, determine the length of the fields that will expand. Next, calculate and add the expansion factor. Then, add the length of the fixed fields.*

2. *Expansion of text contained within line art boxes must also be considered to allow for proper translation.*

3. *The use of abbreviations is not an acceptable way to provide expansion space.*

Example

FILE 'nnnnnnnn' DOES NOT EXIST.

The translatable part of this message (FILE '-' DOES NOT EXIST.) contains 23 characters. According to Figure 3 above, about 70% expansion space should be provided (approximately 17 characters more for translation of this message). Any time a phrase is changed, the expansion factor must be recalculated and reapplied.

Functions Dependent on MRI Expansion

Functions are often invoked by pointing to items listed on the display.

RULE 290:
Functions dependent on display field length and/or position must not be designed in such a way that they are affected by MRI expansion.

Enabling

All products that contain functions dependent on the MRI must be designed so that these functions are not affected by the translation process. Changes in field length and/or position must be possible at translation time.

Example

In the following example, a help function is selected by placing the cursor under any character of the desired field. If expansion, because of translation, is not considered during the design (as can be seen in the example of a German translation below), a cursor placement under the H of HILFE5 will return HELP5. Under the I, it will return an error, and under any of the remaining characters (LFE5) it will return HELP6.

```
HELP1 HELP2 HELP3 HELP4 HELP5 HELP6 HELP7 HELP8
HILFE1 HILFE2 HILFE3 HILFE4 HILFE5 HILFE6 HILFE7 HILFE8
```

Expansion Space for Panels

The more information presented on a screen, the more difficult it is to translate to another language.

Enabling

The calculation of expansion space for panels must include space for input fields and space for constant text (for example, prompts).

For MRI data that is centered on the panel, expansion space on both sides must be provided. Likewise, for column headers, calculated expansion spaces must not only be allowed for but also restricted within the respective columns, in case the translated headers should expand into a new line.

On a menu type panel, if an item is not provided with the expansion space necessary on the same line, a complete extra line must be provided.

Information panels such as tutorial or help panels can be organized like a book that is stored in the computer and read on the display terminal. With this arrangement, the number of information panels can be increased or decreased to allow for the increase or decrease because of translation. Techniques should be available to allow for expansion of information panels beyond the base physical limitation or beyond the expansion space described in the MRI expansion rule.

Example

```
                    *PROBLEM ANALYSIS*

Problem Analysis Options Available: Select Option and Press ENTER

 ID  Choice
 --  ------
  1  Start Problem Analysis Routine
  2  Display Problem Analysis History
  ...
  8  Run Processing Unit Analysis Test
  9  Send IBM Error Information (If Remote Facility Installed)
  ...

 Select = = = > 1
```

Both the header line and item number 9 will need an extra line when translated. It may be necessary to develop two panels, although one would have been sufficient if the panel had been designed to allow expansion space on each line.

Example

Windows designed to CUA specifications do not have to contain any extra space for expansion if there is a means of automatically expanding or shrinking the window to fit the translated MRI.

Chapter 8. Panels and Messages

A panel is an arrangement of information presented to the user. Different types of panels are used to present different kinds and arrangements of information, including menu panels, parameter entry panels, information panels, help panels, data entry panels, text panels, and forms fill-in panels. The use of standard formats is an important factor for the translator because it contributes to the efficiency of the process and the quality of the translation.

A panel consists of a set of display fields and blank space. A display field is a set of characters that can be viewed or operated on as a unit. In the Common User Access (CUA), the display fields are categorized as protected fields, entry (input) fields, and selection fields.

A selection field is a group of choices from which the user can select. A single choice within the field can be in the form of words, an icon, or a combination of them. When the form of the choice includes words, it is subject to translation and must follow the rules for MRI enabling.

An entry field is a space in a panel where the user can enter information. Entry fields can take on three formats. The simplest is a blank field. Depending on the nature of the data to be entered, expansion space may or may not be required. The second format contains a default value. Again, depending on the nature of the data expansion, space may have to be included. The third format contains formatting information that forces the data entered into a pre-established form. The sensitivity of this format to language requirements will be discussed in this and the following chapter.

Panels are an important user interface mechanism and have a major impact on usability. Panels must be designed with consideration for world wide users. Panel design has an impact on the quality of the translation and the productivity of the translator. A mechanism or software tool should be provided to assist the translation process for panels. For example, the translator should be able to view the panel and enter the translation simultaneously.

When panels are designed using development aids such as *EZ-VU* and translation is done using a companion tool, many of the enable rules and guidelines are followed automatically. Even when these tools are used, however, the designer is responsible for ensuring that *all* rules are followed. Even more important is the requirement for the designers of development aids to provide for the rules and guidelines.

Specific application panels such as calendar panels are particularly sensitive to country variations and must be designed with both translatability and the culture differences in mind. The cultural differences are addressed in later sections dealing with date, time, numbers, monetary values, and weights and measures.

Identification and Tracking

RULE 300:
There must be a method provided to allow identification and tracking of panels and messages during the translation process.

Enabling

The translator must have a mechanism for identifying and tracking messages and panels. A commonly used method is to give each message and panel a unique identifier that can be displayed on request and can be used to simplify cross-referencing. Whatever method is used, it must allow easy identification of messages and a cross-reference to panels for ease of translation within the correct context. Where there is a publication associated with the message or panel, the identification could be used to cross-reference between them.

For information panels (tutorial or help panels), the identification method must accommodate the addition of panels that may be necessary because of expansion for translated material.

Denoting Variables and Input Fields

The delimiters for variables are normally displayed to the user and are considered a part of the translatable MRI. Variables themselves are usually provided by the application and are not a part of the translated MRI.

It is important to use a consistent convention for both variables and input fields so that the translator can easily recognize the type of data.

GUIDELINE 1140:

A consistent convention should be used throughout the product for denoting variables and input fields.

The character chosen to denote variables should be taken from the punctuation or special symbols shown in the Syntactic Character Set shown in Figure 1 on page 10. Delimiters in products using ASCII or ISO codes should be selected from the same set of characters as for EBCDIC to maintain compatibility at the system level.

Example

For example, in denoting variables, if % is used to denote a variable such as "%name," then the % should consistently be used in that context.

Location and Order of Variables

The location of variables and input fields within a display field is dependent on the syntax of the national language.

RULE 310:
Variables must be permitted to assume any location and order within a display field.

Enabling

The location and order of variables and input fields within a display field must not be restricted because they can be influenced by the national language sentence structure. It can be necessary to relocate them or to alter their order to comply with the new syntax.

Example

If we look at the message:

> 'retcode' RETURNED BY FUNCTION 'name'.

and a translation of it into German:

> DIE FUNKTION 'name' ENDETE MIT DEM FEHLERCODE 'retcode'.

There are two variables, the name of the function in error and the return code that gives the reason the function had failed. The sequence of these two variables is reversed by the German translation. A literal translation that preserves the sequence of variables would be incorrect.

Message Construction

If messages are constructed (that is, formed from words or phrases) into a sentence or thought, the MRI may be difficult or impossible to translate.

RULE 320:
Messages and other displayed words or phrases must be complete entities and not be constructed from individual words or phrases.

Enabling

A phrase used in two different sentences may require different wording, forms of words, or word gender when translated to another language. Message construction also causes the words in each phrase to be positional within the phrase, thus limiting the translator's ability to place words in their correct grammatical sequence.

Example

The following is an example of combined messages that are appropriate in English but cannot be translated consistently into another language. The first table shows the English nouns (on the left) and the adjectives (on the right) that might be combined to make up a single message.

Terminal	*operational*
Control Unit	off line

The result of two of these combinations would be:

Terminal *operational*

Control Unit *operational*

In French, however, the different genders of the French words for terminal and control unit require different forms of the adjectives.

Terminal *opérationnel*

Unité de controle *opérationnelle*

Notice the two forms of the French adjective for the single English adjective *operational.*

Similarly, words themselves should not be constructed from word fragments. The word "DAY" could be joined with the word fragments "MON," "TUES," "WEDNES," "THURS," "FRI," "SAT" and "SUN" to form the days of the week. However, there are no other languages in which this construction will work. For example:

FRENCH	GERMAN	ENGLISH
LUNDI	MONTAG	MONDAY
MARDI	DIENSTAG	TUESDAY
MERCREDI	MITTWOCH	WEDNESDAY
JEUDI	DONNERSTAG	THURSDAY
VENDREDI	FREITAG	FRIDAY
SAMEDI	SAMSTAG	SATURDAY
DIMANCHE	SONNTAG	SUNDAY

The French suffix "DI" seems to work like the English "DAY" until you come to Sunday (DIMANCHE) and the German suffix "TAG" is not valid for one day (MITTWOCH).

Reuse of Messages

Messages with double meaning can be difficult or impossible to translate.

Enabling

It is sometimes possible to use a single message under two different circumstances because of the dual meaning of words. However, when the message is translated, it may be necessary to create two distinct messages.

Example

RECORD ERROR. (Meaning, "There is an *error* in this *record.*")

and

RECORD ERROR. (Meaning, "*Record* this *error* in the log.")

However, when translated, different phrases must be used to reflect the correct meanings. In French, they translate as:

CET ENREGISTREMENT COMPORTE UNE ERREUR

and

VEUILLEZ NOTER CETTE ERREUR

In addition to messages, there are other types of display field contents that, for the same reason, must not be reused if different meanings are implied.

End-User Commands, Keywords, and Responses

All end-user commands, keywords, and responses are part of MRI and as such follow the MRI rules and guidelines.

Accepting End-User Commands, Keywords, and Responses

End-user commands, keywords, and responses, like messages, must be translatable and therefore must not be hard-coded. End users must be able to enter commands and responses in their national language or in a form compatible to their national language.

RULE 330:
End-user commands, keywords, and responses must be treated like other MRI.

Enabling

End-user commands (for example, ERASE), keywords, and responses (for example, yes/no) must be treated as with the following considerations.

1. End-user commands must be distinguished from instructions in various programming languages, such as COBOL and FORTRAN, which are not normally translated.

2. If national languages are used for commands (or NL synonyms or mnemonics) in end-user languages, such as REXX or QMF, the interpreter or compiler needs to have an English version of the command concurrently active to allow portability of the resulting programs.

3. National languages must always be usable for compiler source comments, because they are not compiled.

4. Like other MRI, the end-user commands, keywords, and responses need to be isolated at the source and packaging level from the associated executable code, as in VM/SP Release 5 command files. These facilitate command identifying and locating for translation.

In some cases, commands, keywords, and responses allow abbreviations or mnemonics for ease of use. The form of the abbreviation or mnemonics should not be dependent on the first letters or on the mnemonics of the English versions. As well, the ability to abbreviate in all national languages must not be assumed. For some national languages, different commands start with a similar sequence of letters, making abbreviation difficult if not impossible. The abbreviated form of the English should not be used as a basis for calculating the expansion factor for translation.

The CUA presents rules for selecting mnemonics of choices and commands. These are generally applied to non-ideographic languages as well as to English.

The developer must also be aware that a translation of one word can result in more than one word separated by a space. The implications of this possibility are discussed in "Control Code Points for Software" on page 15.

Example

A typical response is the word *Yes*. In Italian and Spanish *Si* should be used, for German and Dutch *Ja* and for French *Oui* is the answer. Each of these can have an alternative response. One method of implementation uses *equivalences*. For example, responses are contained in a separate file and are therefore available for translation. This file could contain all the other responses that the application could accept, in addition to the standard response. In this way a *YES, Yes, Y, JA,* or *Ja,* could be allowed, although the number of acceptable answers could be very large. It is not acceptable to use 1 and 0 to indicate YES and NO responses, even if each question specifies the translated equivalents. One reason is that if in a given language, 1 normally signifies NO, a question asking for 1 to signify YES would result in confusion. A further reason is that some countries do not commonly use Arabic numerals.

Use of Cases in Commands, Keywords, and Responses

Not all products are translated into the national language of a country. Some licensed programs, particularly command languages, are left in their English form when installed in other countries. Some national language character sets do not contain the lowercase of the English alphabet. Katakana and Thai are good examples. Commands cannot be entered in lowercase on products supporting these character sets.

RULE 340:
Entry of end-user commands, keywords, or responses must be possible without regard to uppercase or lowercase.

Chapter 9. National Usage Considerations

Any function of a product that must be altered to suit a new country or language is defined as a National Language Dependent Function (NLDF). These functions may also vary by language within a country, such as French and English in Canada. NLDF is distinct from LF (Linguistic Function). NLDFs typically consist of a common core function or algorithm that is modified with a small table for each environment. One installation might have several SBCS national language tables available at the same time for each NLDF. Examples of NLDFs are sorting, searching, keystroke processing, and calendar formats.

In contrast, an LF typically has little in common between different language implementations. LFs are dictionary-oriented and rather large, so a new national language is a major undertaking. Examples of LFs are hyphenation, spell checking, text-to-speech synthesis, and speech recognition.

For specific information on national language considerations for some countries, consult the other volumes of this guide.

Designing for National Language Dependent Functions

RULE 350:
A product with National Language Dependent Functions must be designed to facilitate addition of other countries or national languages.

Enabling

The product must be designed so that additional countries or languages can be supported without redesigning the base program code.

Sort, search, and collate functions have different rules in different countries and in different languages. The sorting order must be changeable without modifying the executable code.

Some LFs require additional memory for execution and additional space on the packaging media for code, to include an expanded national language version.

Example

One design method is to provide an exit at each of the country or language dependent routines. (See Rule 130 in "National Language Exits" on page 21.)

Example

Each product that uses a sort or related function must have the flexibility of allowing the user to define the desired collating sequence. This can take the form of a choice of predetermined tables, for example, English, German, Swedish/Finish, or Danish/Norwegian. Many languages have characters (many of which are common usage vowels essential to the correct spelling of a word) outside the common, unaccented, A–Z alphabet. These characters must be considered for collating purposes. For example, a display of a simple list of documents on a diskette, if allowed to default to the EBCDIC code point order, will only appear in alphabetic order for the English user. A European user will be presented with an apparently random order because most of their document names will contain national characters that are not in any order in the EBCDIC tables. This user will be penalized with greatly reduced usability for using a national language.

Example

Do not collate or sort on the basis of the hexadecimal value of the internal representation of the characters being manipulated. Many languages have non-English alphabetic characters that must be considered. In general, the hexadecimal value of their code points has no relationship to their sort sequence.

Example

Some languages consider a group of two or three letters as if it were a single letter in a sorting process. For example, in Spanish the word *llama* has the double *ll* evaluated as a single character. Similarly, in Catalan, which is spoken in the north east of Spain, the three consecutive letters *l.l* (the dot is normally placed at the center at hyphen level) that appear in words such as *carel.li* are evaluated also as a single character.

Example

The following names include character strings that are ignored in some countries when sorting alphabetically:

Bertha van der Vorst
Franz van Essen
Francesca d'Allessio

Umberto del Zoppo

The following is the result of sorting the above list according to the last name:

d'Allessio, Francesca	(sorted as Allessio)
van Essen, Franz	(sorted as Essen)
van der Vorst, Bertha	(sorted as Vorst)
del Zoppo, Umberto	(sorted as Zoppo)

Note that the articles (d', van, van der, and del) are ignored for the sort.

Example

LFs and NLDFs must not be bundled with other functions in hardware and software that inhibit NLS. Voice systems are naturally associated with telephony, and telephone functions require homologation. Thus, a voice card that embeds dialing and other telephone functions may not be legally used in other countries. The telephony functions could be designed for easy removal or alteration to enable the voice functions.

Date and Time Conventions

Date and time conventions vary from country to country. The following is a list of some calendar systems used in different countries:

Julian Calendar	pre-Gregorian calendar.
Gregorian Calendar	current calendar used in the U.S.A. since 1752.
Julian Day	astronomical day count. Jan 1 1987 is Julian day 2,446,795.5.
Lunar Calendars	used in various countries.
Day number reference	YYDDD. June 10, 1987 is 87161.

RULE 360:
Date and time formats must be selectable.

Date Format

Enabling

Date format is not standard worldwide.

Example

Some commonly used formats are shown in Figure 4. Formats must be user-selectable.

YY/MM/DD	YY.MM.DD
YYYY/MM/DD	MM-DD-YY
YY-MM-DD	MM/DD/YY
DD/MM/YY	YYDDD
DD.MM.YY	

(DD = day, MM = month, YY = year)

Figure 4. Date Formats

There are other formats in use, but if the format is selectable by the user, all formats can be accommodated.

Note: In the U.K., it is common letter writing practice to add *st., nd.,* and *rd.*, after the day number (for example; 1st., 2nd., 3rd., 21st.). This practice should be considered when automatic date insertion is planned for a letter or when allocating space for an input field.

Time Format

Enabling

Time format is not standard worldwide.

Example

Some commonly used formats are shown below.

24 Hour Convention	HH:MM:SS
	where
	HH = 00-23
	MM = 00-59
	SS = 00-59

12 Hour Convention	A.M, P.M. HH:MM:SS
	where
	HH = 00-12
	MM = 00-59
	SS = 00-59

Figure 5. Time Formats

Numbers

Use of period and comma in numbers vary from country to country.

RULE 370:
Numeric punctuation must be selectable.

Enabling

There are three basic styles of period and comma conventions required to satisfy national language standards for numerics.

Example

Typical conventions for number formats are:

Decimal Convention	Example	Example	Example
Period	3,872,345.96	7,456.54	.783
French	3 872 345,96	7 456,54	0,783
Comma	3.872.345,96	7.456,54	0,783

Figure 6. Numbering Conventions

Number Rounding and Mathematical Formats

RULE 380:
Number rounding and mathematical formats must be selectable.

Enabling

Countries use different rounding rules and different number presentations in their applications. A user must be able to specify the usage of these during operation and also change their default values.

- Specification of automatic insertion of negative number signs like hyphen, brackets, square brackets, and greater/less than brackets.

- Several conventions are in effect for the recognition of negative numbers. The common U.S.A. version of a leading hyphen is not widely recognized in Europe.

Example

Conventions for negative numbers are:

– Parentheses	(10)
– Square Brackets	[10]
– Leading Hyphen	-10
– Trailing Hyphen	10-

- Attention must be paid when whole negative numbers have to line up on defined decimal tab settings. Frequently, examples are shown that have decimal places and are surrounded by negative brackets. Because the right-hand bracket is to the right of the decimal tab stop, everything is in order. However, in many accounting listings, the decimal places are omitted because of the large monetary values involved and whole negative numbers are bracketed. In this case, the right bracket is typed prior to the decimal position being reached (in fact, there is no decimal) and is therefore incorrectly aligned under the units position. A method must be supplied that enables exiting from the decimal mode, or the right bracket must be treated as a nonescaping character in this case.

- Specification of text to print in either a leading or trailing position around positive or negative numbers such as *debit, credit*, or the corresponding national equivalents.

Use of Words

<u>*GUIDELINE 1150:*</u>

Words should not be used in place of numbers.

Example

A billion in many European countries is:

```
1,000,000,000,000
```

but a billion in the U.S.A. is:

```
1,000,000,000
```

Monetary Values

RULE 390:
Monetary format must be definable.

Example

Some different monetary formats are:

1,234.56	(U.S. Dollar)
1.234,56	(German Mark)
1 234,56	(French Francs)

Some Arab countries use three digits after the decimal separators. Note that a leading zero must be included for numbers whose absolute value is less than one. Note also that in the French decimal comma convention, the space at the thousands position may be suppressed for numbers having only four digits to the left of the decimal comma.

Currency Symbol

RULE 400:
The default currency symbol or its abbreviations must be selectable.

Enabling

Currency symbols (for example, $) must not be hard coded and must be changeable by the translator. For some countries, this currency symbol abbreviation requires three characters (for example, Pts for Pesetas). The three-character currency abbreviations of ISO 3166 are recommended.

Position of Currency Symbol

RULE 410:
The currency symbol position must be selectable.

Enabling

Some countries place the currency symbol at the end of the value, and some even put it inside the value.

Example

$123.45	(U.S. Dollar)
DM123,45	(German Mark)
123,45 F	(French Francs)
123$45 ESC	(Portuguese Escudo)

Space for Monetary Value

RULE 420:
Field sizes for monetary values must be selectable.

Enabling

The field size must allow for the larger values common in many countries.

Example

Currently, a value expressed in Japanese Yen require two more significant digits for the same relative value expressed in U.S. dollars.

Weights and Measures

Most countries use the metric rather than the English system commonly used in the U.S.A.

RULE 430:
The measurement system must be selectable.

Enabling

Weights and measurements must be definable in the measurement system most familiar to the end user.

Example

A country may be using several measurement systems. For example, the U.K. has changed to a metric system of weights and measures, but gasoline is still most commonly purchased in imperial gallons, and all road signs show distances in miles.

Chapter 10. Terminology—Grammatical and Style Sensitivity

This chapter suggests some guidelines appropriate for technical writing that may be translated into other languages.

Consistency

GUIDELINE 1160:

The terminology used in MRI should be consistent throughout a product.

If variations in terminology are used in the MRI, translators will waste time trying to determine if a different, more appropriate, word is required. The following examples illustrate unnecessary variations in terminology that translators have found:

- Workshop, plant, factory, facility
- Core, main memory, RAM.

Abbreviations

GUIDELINE 1170:

Abbreviations should be avoided.

Abbreviations lead to misunderstandings by the translator and by the end user.

Rules for abbreviation of words vary from language to language. Do not assume that translators will understand the meaning of your abbreviations or be able to abbreviate their translations. Some languages (for example, German) allow only limited abbreviations, and other languages (for example, Arabic) do not allow abbreviations at all.

Do not abbreviate an English word to create expansion space for translation. The target language may not be able to use abbreviations and will still require the additional expansion space. For example, the character "#" used in the

U.S.A. as an abbreviation for number does not have this meaning in other languages.

If abbreviations cannot be avoided, it is strongly recommended that only common abbreviations be used and that all abbreviations be explained to the translator. In any case, all MRI rules and guidelines still apply.

Do not use cryptic messages, such as:

NON-ZERO RC FROM AA1DBMPX EXEC

A great deal of expansion space will be required to make an acceptable translation of such a message.

Mnemonics create their own problems. Although mnemonics are recommended for use in choice selection situations where selection menus are presented, the mnemonic itself must be changeable to match the translated selection menu.

Slang

GUIDELINE 1180:

Slang, jargon, and humor should not be used.

Slang is never appropriate and can even be offensive if misunderstood. Jargon is often coined locally and also may not be understood. Furthermore, jargon seldom can be found in dictionaries or reference manuals if the translator needs understanding or clarification.

Trademarks

GUIDELINE 1190:

Trademarks should be identified and explained.

Observe the conventions for trademarks. As well as being necessary for legal reasons, the "circle R" symbol or the use of a footnote will indicate to the translator that the word is not likely to be found in any ordinary dictionary. "Fiberglas," a trademark of the Owens-Corning Fiberglas Corporation, for example, is a word that may not appear in a translator's dictionary.

Ambiguous Words

<u>*GUIDELINE 1200:*</u>

Ambiguous words should not be used.

Avoid ambiguous words when designing MRI. For assistance in selection of word and sentence style, see IBM publications *Dictionary of Computing,* SC20-1699.

Sentence Structure

<u>*GUIDELINE 1210:*</u>

Proper style and sentence structure should be used in MRI.

The following is a list of recommendations:

- Use simple declarative sentences whenever possible.
- Do not use language that might be misunderstood and incorrectly translated.
- Be careful in the use of compound phrases such as "card input." Translators report that the phrase "card input" has been used in the same documentation to mean "the input of a card," "the input of several cards," or "the concept of card input generally." Similarly, the phrase "program definition" has been used with the meanings "definition OF the program" and "definition BY the program."
- Avoid compound adjectives. Multiple-word adjectives cause translators to guess where the adjective stops and where the noun starts.
- Do not make sentences too long. A long sentence is difficult to translate, especially if the sentence contains several ideas or concepts. Sentences need not be artificially short, but they should have a simple structure.
- Do not construct sentences that embody a series of concepts separated by commas. It is much easier to understand such material when it is presented as a list.
- Check typographical errors. To the translator, a simple typographical error can cause a disproportionate amount of trouble. Unless the error is obvious, the translator may think that the misspelled word is a new word. Use automated spelling tools.

- Do not omit prepositions or articles from sentences in an attempt to save space. Avoid telegraphic style. Do not try to save space for expansion by truncating a sentence.

Example

The following can be interpreted in different ways.

"ADDRESS MATCH COMPARE STOP DETECTED"

1. *The* ADDRESS *of the* MATCH COMPARE STOP *has been* DETECTED.
2. *An* ADDRESS MATCH *of the* COMPARE STOP *has been* DETECTED.
3. *An* ADDRESS MATCH *has occurred so* COMPARE *the* STOP DETECTED *with what you wanted..*

Negative Questions

GUIDELINE 1220:

Negative questions should be avoided.

Negative questions are quite often misunderstood by the user. In addition, the correspondence between "yes" and "no" and equivalent national language words is not the same for negative questions as it is with positive questions.

Example

A positive response to a negative question in French is not "oui" but "si."

Part 3: Bidirectional Subset

Some languages are different from Latin based languages in more than character sets alone. For example, the general direction of Hebrew and Arabic text is from right to left across the page in successive lines from top to bottom down the page. However, numbers and Latin character phrases are written from left to right. As text and numbers are often intermixed, these languages are bidirectional in nature. These bidirectional languages require additional enabling rules and guidelines to ensure that they can be implemented. The requirements for special keystroke handling and data manipulation of these languages put unique demands on the developer of an enabled product.

Chapter 11. Enabling for Bidirectional Support

The left-to-right orientation of Latin based languages is inherent in the design of most IBM products. However, the right-to-left orientation also exists, and any architecture or proposed method of implementation must not preclude this other orientation. Some languages are written in a generally right-to-left orientation with certain elements written in a left-to-right orientation. They are bidirectional in nature.

General Design Consideration

It must be stressed that Chapter 4, "General Design Considerations," as well as the rest of the base subset, applies equally to the Bidirectional Subset. The modularity of the design of hardware, microcode, and software must take into consideration the rules and guidelines of this subset. Exits must be planned at strategic points for the correct handling of all the functions of the Bidirectional Subset and their pervasive effect on I/O operations.

To be enabled, a product must allow the implementation of a new language, including those covered by the Bidirectional Subset, without redesign or retrofit.

Logical Order of Data

RULE 440:
Hardware and system software design must allow for bidirectional data to be passed to applications in the same order a speaker of the language would spell it out.

Arabic and Hebrew are typical bidirectional languages. The text of these languages is written from right-to-left, but numeric data is written from left-to-right. In addition, there is a need to imbed Latin-based text, such as English (as in the case of a company's name), within the Hebrew or Arabic text. Bidirectional controls, therefore, are required to provide adequate language support for Arabic and Hebrew.

Bidirectional Functions

Products must be designed in such a way that these controls and their functions can be implemented.

RULE 450:
The product design must allow for the implementation of the correct handling of bidirectional keyboard and presentation functions.

Enabling

Bidirectional functions, depending on the situation, may have to be handled by software, microcode or hardware, singly or in combination. When the keyboard is involved, it must allow for the handling of bidirectional operations.

Bidirectional functions include:

- Selection of the default view-port orientation, where view-port stands for document, page, screen, window, or any other presentation set suggested by the environment
- Symmetric swapping of directional characters and its activation
- Field Orientation.

In the following list, the term "field" is used generically to refer to the smallest character grouping supported by a given environment, such as a field, a line, or a sentence.

  - Setting of the orientation of a field
  - Inversion of field orientation
  - Nesting: Temporary reversal of the flow of data within a field (or line, or sentence).

Enabling

Although these functions do not have to be actually implemented for the product to be enabled, the designer should understand the functions enough to grasp the impact they have on the product being designed. Appropriate exits and memory to contain the extra code must be considered in developing an enabled product.

Selection of Default View-port Orientation

In this section, the term "view-port" is used generically to refer to document, page, screen, window, or any other presentation set suggested by the environment. These presentation sets may, in some cases, be related hierarchically. For example, a default orientation may be selected for a document and another for a page. In such a case, the page orientation overrides, within its boundaries, the default set for the document.

It is necessary to know and control the orientation of the view-port to allow the data to flow in the correct direction. Furthermore, when a mirrored keyboard function has been implemented ("Mirrored Keyboard Functions" on page 73), there is a need to know which orientation to use as the default. This default is set according to the view-port orientation. When no view-port orientation has been selected, the default must be left-to-right. The design must allow the implementation of a means of selecting the view-port orientation.

Example

When the tab key, a mirrored function, is pressed, the view-port orientation will determine whether the next tab stop is on the right or on the left.

Symmetric Swapping of Directional Characters

Certain characters have a meaning attached to their direction. For example, the character left parenthesis "(" also has the meaning of *open parenthesis* in left-to-right languages and will be generally treated as such by most products. This very character, though, is a *close parenthesis* in Arabic or Hebrew.

This "open/close" or "begin/end" meaning attached to pairs of directional characters is symmetrically reversed in Arabic or Hebrew. To retain the meaning rather than the direction of those pairs of characters, provide the facility to encode the meaning - open parenthesis - and display it as either a left parenthesis in a left-to-right context or a right parenthesis in a right-to-left context.

The "open" or "begin" meaning must be attached to the code point associated with the "left" element of the pair and the "close" or "end" meaning to the "right" element. When the current orientation is left-to-right, those characters will be displayed as they are in English, while they will be swapped with the symmetric member of the pair when the current orientation is right-to-left.

The design must allow the implementation of symmetric swapping and of a facility, accessible to the software only, that allows activation or deactivation of the swapping function. Symmetric swapping must not be controlled by the person using the keyboard.

The appropriate country standards authority should be consulted to obtain a list of the character pairs involved.

Field Orientation

In this section, the term "field" is used generically to refer to the smallest character grouping supported by a given environment; it could be a field as such or a line or a sentence.

Setting the Field Orientation

Within a screen or a page or any other grouping of fields, some fields may require an orientation different from that of the screen or page as a whole. It must be possible to implement a means of setting the special orientation of the field.

Example

In a data entry panel, the screen as a whole may be right-to-left, with the titles, explanatory text, following the default orientation. The fields would be sequenced from right-to-left and top to bottom. Some of the fields, though, may be numeric and therefore require a left-to-right orientation.

Inverting the Field Orientation

It is, at times, necessary for a user to invert the orientation of a field from the keyboard. A means to produce this result must be implementable.

Example

In a name-and-address-file update panel, the "NAME" field is likely to be set as right-to-left in the Middle East. When the user has to occasionally enter the name of a correspondent in the U.S.A., for example, there has to be a facility to invert the field orientation for this particular entry.

Nesting

It is necessary to be able to reverse temporarily the flow of data within a field. Such reversal is called nesting. Provision should be made to allow the implementation of at least one level of nesting (reversal from and return to the basic field orientation) in all cases, and at least two levels (reversal within a reversed segment) when the data stream allows in-stream controls (controls intermixed with data).

Example

One level of nesting is necessary to allow entry of numbers within Arabic or Hebrew text, as in an address. The field orientation is right-to-left, the street name is entered from right-to-left. Then the flow has to be reversed to allow correct entry of the number from left-to-right.

When possible, when in-stream controls are supported as would be the case in a text application, nesting should be permitted to two levels at least. This nesting would allow the entry, for example, of a right-to-left address (street and number) into a left-to-right sentence.

Bidirectional Controls in Software

The extra functions and controls that are supplied in hardware for bidirectional control are accessed both from the keyboard and through program control. The software driving these functions must allow for their uniqueness and provide access to them. Even the data streams have unique qualities that must be considered.

The software must also provide the ability to set up bidirectional controls when the attributes of screens and fields are set. In addition, any provision in hardware for dynamic manipulation of these bidirectional controls must be reflected in the software.

MRI and Bidirectional Presentation Controls

As Rule 270 in "Isolation of Source Level MRI" on page 39 mandates, presentation control information must be treated as MRI. This point has to be stressed again for the bidirectional subset. In addition to the familiar presentation controls such as highlight, underscore, and numeric field, all the controls necessary to effect the functions mentioned under the "Bidirectional Functions" heading must be treated as MRI and subject to MRI rules.

Mirrored Keyboard Functions

Keyboards used for monodirectional languages usually include several keys that control the movement of the cursor in an explicit direction (physical movement).

The cursor movement key pairs right/left and up/down will work equally well in a bidirectional environment and are not concerned with mirroring.

Keys that cause logical movement of cursor or characters, such as tab, backtab, backspace (backspace-delete), insert, carriage return, and delete, must have their function mirrored when used in a right-to-left environment. Mirroring means that the implicit movement caused by the operation is reversed when the current orientation is reversed. In other words, mirrored keys will perform as standard left-to-right keys for entry of English text, and with mirrored symmetry for entry of Hebrew or Arabic text.

RULE 460:
Designing of a function that implies logical movement of cursor or characters must permit mirroring of that function.

Enabling

Mirroring must not exclude the original function, because both are needed to support the bidirectional languages.

Example

In English, when a display is in *insert* mode, the data is pushed ahead of the cursor to the right. For bidirectional machines, this direction is correct for left-to-right orientation, but in right-to-left orientation, the data must be pushed to the left of the cursor.

Note: Keys used to control the handling of bidirectional presentation, for example, the key that invokes field inversion, may cause cursor movement. Even though this movement may be relative to the *field* orientation, it is inherent to the key's effect on presentation and must not be treated as a case of mirroring.

Bidirectional Graphic Symbols and Icons

Graphic symbols and icons can present a problem in a bidirectional display if they are directional.

RULE 470:
Keys or operations labeled with directional icons or symbols must perform according to the icon or symbol.

Example

The carriage return (new line) symbol may appear on the display as follows.

```
                  *
                  *
                  *
  *               *
 *                *
* * * * * * * * * *
 *
  *
```

The design of this symbol implies a line feed and a return to the left. In bidirectional machines, the return may be to the left or right depending on the orientation of the setting of the bidirectional controls. Therefore, this symbol must not be used for this function. A word in the national language should be used instead.

This rule also applies to symbols used for tab, backtab, backspace, rubout, and others.

Keyboard Nomenclature

Keyboard nomenclature should be treated like the graphic symbols and icons. Care should be taken in choosing keytop nomenclature that is not limited to a particular direction.

RULE 480:
Keyboard nomenclature for mirrored functions must be independent of the direction of data or text entry.

Example

The example of the carriage return in "Bidirectional Graphic Symbols and Icons" is a case in point.

Inverse Move

An Inverse Move instruction is called for where it is desired to reverse the order of a string. It may be used when building reverse indices, for example, or in sort application where the order depends partly on the reversed sequence of a part of the string (French dictionary order).

The Inverse Move instruction is particularly useful for coders implementing bidirectional systems. It is similar to a regular move instruction except that the resulting character string is the mirror image of the original.

RULE 490:
An inverse move instruction must be part of each processor's instruction set.

Example

The following diagram shows the effect of the Inverse Move instruction. S is the source field, and D is the destination field. The double quotation marks delimit the string and are not part of the string. (The format shown here is for illustration purposes only.)

S = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

D = "XXXXXXXXXXXXXXXXXXXXXXXXXX"

Applying the Inverse Move instruction with 6 as the length.

```
IMOV  D, S, 6
```

The resulting field D is as follows:

D = "FEDCBAXXXXXXXXXXXXXXXXXXXX"

This instruction is useful in controller microcode for both displays and printers.

Independence of Display Functions

In some displays, it is convenient to "blank to the end of the line," "blink to the end of field" or "something to the *end* of something." The designers mean the right hand end of whatever is being defined. In bidirectional languages, the *end* of the line or field may be at either end depending on the nature of the data or text on the line. The end of something may not always be the rightmost part of it.

RULE 500:
Display functions must not assume a left-to-right orientation.

Field Attributes

In data-oriented systems, the display functions are controlled by field attributes. To trigger the direction-sensitive aspect of these functions, the field attributes must carry the direction information.

RULE 510:
Field attributes must contain room for directional information.

Reserved Bidirectional Indicators

As suggested for language shift indicators in Chapter 6, "Keyboard and Keystroke Processing," bidirectional indicators must be reserved.

RULE 520:
Indicator location must be reserved for the current direction of the cursor (direction of input).

GUIDELINE 1230:

Indicator locations should be reserved for screen and field orientation, current level of nesting, status of push (nesting mechanism), and status of symmetric swapping.

GUIDELINE 1240:

The design should provide for a method to indicate to the user the nesting structure of a string.

Bidirectional Text in a Graphic Environment

It is common in graphic environments, whether business graphic, technical design, free-hand drawing, image processing, and others, to complement the "art work" with text for labelling, explanation, legend. There are several reasons (for example, aesthetic selection of the mirror image of a picture) to treat text and "art work" separately with regard to their orientation (direction of flow for text, origin for graphs).

This independent orientation of text and graphic becomes more important in a bidirectional environment. Graphics in the bidirectional world often have a general right-to-left orientation, in other words, the origin of the horizontal axis is to the right with the weight of the values progressing from right to left. It must be possible to produce such a graphic for local use and be able to reverse (mirror) its orientation for international consumption and vice versa. This graphic orientation reversal should not affect the orientation of the bidirectional text that complements it.

RULE 530:
The design must allow for independent handling of graphic and text orientation.

Chapter 12. Special Considerations for Arabic

In addition to bidirectional controls, the Arabic language has other unique requirements.

Shape Determination

In Arabic and related languages, the shape of a letter in a word changes, depending on which letter precedes and succeeds it, and on its own ability to connect. Characters may have *initial, middle, final,* and *isolated* forms, but not all characters have all forms.

On the keytops of an Arabic keyboard, only one shape per letter is represented, while all the shapes can be coded using the Arabic code page and must be available for presentation. For efficient processing, it is necessary to use only one code point per letter, while for presentation the proper shape in context must be selected. As a result of these constraints, a shape determination routine is required for Arabic and the related languages.

RULE 540:
Provision must be made to allow shape determination to be performed.

Enabling

The design must provide the appropriate entry point to allow a shape determination routine to be executed. In addition, there must be enough memory to contain the shape determination routine.

The shape determination routine must allow for automatic (algorithmic) selection of the appropriate shape according to the context when so directed by the software or the user. It must also allow for user or software controlled selection of any of the four shapes mentioned above as well as the "base" shape and the "keytop" shape. The routine must also allow transparent pass through of data, in other words become temporarily deactivated, under software or user control.

Conversion to Base Shape

It is expected that the data intended for processing will always be passed in *base* shape (that is, one code point per letter regardless of the shape) to the application. When, by exception, shaped characters are to be handled by an application, a system-wide method for converting the shaped characters (or character string) to the appropriate base code points should be provided. In this document, this process will be called *deshaping*.

GUIDELINE 1250:

A system wide method of deshaping Arabic characters or character strings should be provided.

Shape Sets to Base Shape Definition

The application must be supplied with a method of defining to the deshaping routine the base shape code points and the set of shape code points associated with each base shape.

RULE 550:
The deshaping must be definable.

Status Indicator for Shape Determination

When appropriate, a status indicator reflecting the current mode of operation (contextual selection, pass through, specific shape selection) of shape determination should be provided.

GUIDELINE 1260:

An indicator location should be provided for the status of shape determination.

Arabic and Hindi Shapes for Numerals

Most languages use the Arabic shapes for the numerals 0 through 9. However, parts of the Arab world use Hindi shapes for them. There is obviously a one-to-one correspondence between the Hindi and Arabic shapes used to present numerals. Both sets of shapes can be coded using the Arabic code page.

RULE 560:
Provision must be made to allow the selection of the appropriate presentation shape for the numerals.

Enabling

For efficient processing, the internal code points associated with the numerals must, regardless of the shapes selected for presentation, match the code points considered numeric by the environment, that are the code points accepted as digits by the instruction set. In other words, the internal code point associated with the digit 5 must always, in an EBCDIC environment, be coded as X'F5' regardless of whether its Arabic or its Hindi shape is selected for presentation.

Generation of the Hindi shapes or the Arabic shapes from numeric code points, according to the active mode of presentation, is necessary to support Arabic.

A method of overriding the active mode, by the user or the software, must also be provided. In practice, Arabic and Hindi shapes will not be mixed in a numeric field. However, it is necessary to show both Arabic and Hindi numeral shapes in different fields or positions on the same screen or page. This requirement only affects the presentation and appearance of the numerals and should not affect decimal operations.

Entry of a digit from a keyboard (or its numeric key pad) that shows both the Arabic and the Hindi shapes in its two language layers must result in the production of the same numeric code point for the host and its applications regardless of the shape set currently selected for presentation.

Cursive or Touching Characters

Cursiveness is the quality whereby two adjacent characters touch each other or are connected. This is required to make the Arabic characters readable, because Arabic characters are connected.

RULE 570:
Characters must be allowed to touch each other on printers and displays.

Enabling

This rule applies to displays and electronic font printers such as laser printers, matrix printers, and ink jet printers.

Hard font printers must also allow font cells to touch on paper to avoid gaps between letters. Some examples of hard font printers are daisy-wheel printers and chain or belt printers.

Proportional Spacing

Because Arabic, Farsi, and Urdu are cursive script languages and because there is a marked difference between the widths of the letters, proportional spacing for printers is essential.

GUIDELINE 1270:

Printers should be designed so that proportional spacing can be provided.

Character widths vary from as little as the English *i* to considerably wider than the *W*. In present implementations, some characters must be assigned two character cells to be represented adequately with a monospaced font. Any shape determination algorithm must take this situation into account. Proportional spacing on displays is also desirable.

Whenever a printer or display implements proportional spacing, a fixed pitch mode should also be available to allow easy presentation of tabular data.

Baseline Alignment for Mixed Text

Arabic characters have much larger descenders than Latin characters. Consequently, the baseline of standard Arabic fonts is higher than that of standard Latin fonts. Because of this marked difference, text mixing Arabic and English appears uneven. This situation is also true of "Arabic" numerals that appear somewhat below the general baseline when used with Arabic script as is the norm in several Arabic speaking countries.

GUIDELINE 1280:

A method should be provided to allow alignment of the baseline of Arabic and Latin characters (including "Arabic" numerals).

Part 4: Double-Byte Characters Subset

The written language of Chinese and part of Japanese and Korean is ideographic. It uses a different character for each different concept. These scripts require unique characters numbering in the thousands for the most basic information processing. A method is needed to represent these characters that extends beyond the 256 possibilities a single byte provides. Using 2 bytes to represent these characters extends the range to over 64 000 possibilities.

Chapter 13. Double-Byte Character Set

National languages that use ideographic characters require several thousand characters as a minimum character set. A coding system using 2 bytes was developed to represent these characters correctly.

Graphic Code Points for Hardware

Careful distinction should be made between the control codes and the graphic codes. In double-byte mode, coded graphic characters must be represented by codes in a specific range.

RULE 580:
DBCS code points in the graphic character range must be used only for graphic characters and must not be used for control purposes.

Note: Graphics character ranges for some representative code pages (where HOST is IBM S/370, PC is IBM PC, and X indicates a hexadecimal value):

```
(DBCS-HOST)         X'4040' = DBCS SPACE

               X'yyzz' where
                       41 <= yy <= FE  and
                       41 <= zz <= FE

(DBCS-PC)   X'8140' = DBCS SPACE

               X'yyzz' where
                       81 <= yy <= FC
                       40 <= zz <= 7E  and
                       80 <= zz <= FC
```

Single-Byte Meaning

When dealing with a character string that may contain DBCS coded characters, any interrogation must always keep track of whether the information being interrogated is part of SBCS information or DBCS information.

RULE 590:
Single-byte meaning must not be drawn from either byte of double-byte data.

Example

The DBCS character X'457D', which means "beauty," when using the Kanji Host (IBM S/370) code page, has as its 2nd byte the value X'7D', which is an SBCS *single quotation mark* ('). In DBCS mode, the 2nd byte of X'457D' must not be interpreted as a delimiter; rather, the 2 bytes of X'457D' must be considered as a whole, not as 2 single (unrelated) bytes.

Character Generator Size

Depending on the coding scheme used, up to 36 101 graphic characters can be accessed on currently defined code pages. Application portability is enhanced if all character generators are capable of displaying all the characters.

RULE 600:
DBCS character generators must be capable of producing at least 36 101 user-accessible graphic characters, excluding indicator characters and special icons.

Enabling

Although the design of the character generator must enable up to 36 101 characters to be accessed, the actual number of characters that are implemented depends on the language. Presently, the standard defined for the traditional Chinese character set contains 13 728 characters.

Special icons and indicator symbols are not considered part of the language character set. They are produced by the system or application to alert the user to a particular condition. Therefore, their images should be generated separately from any language character. They may, however, for convenience, reside in the display or printer's character generator.

User-definable Characters

The number of characters provided in the DBCS for each language varies. There is always a need, however, for additional characters for a given language depending on the user's application requirements.

RULE 610:
Additional DBCS characters must be definable by the operator, a user, or an application.

Enabling

The DBCS code scheme has set certain code point ranges for user-definable characters in addition to the characters defined in the standard Japanese, Korean, and traditional and simplified Chinese character sets. Part of the character generator must be loadable and dynamically changeable to enable DBCS languages. Although this rule applies primarily to the hardware design, it also applies to software products designed for the creation and handling of character fonts.

Character Cell Matrix

It is recommended that the character cell for DBCS exceed the 24 by 24 pixel minimum requirement when possible to improve readability.

GUIDELINE 1290:

For DBCS, the matrix of pixels forming a character should not be less than 24 pixels wide and not less than 24 pixels high.

Switching between and Coexistence of SBCS and DBCS

DBCS systems do not work in purely double-byte mode all the time. It is necessary to support mixed-byte character strings that contain both single-byte coded characters and double-byte coded characters. The mixed-byte character string can represent a single variable that contains both single-byte and double-byte characters.

RULE 620:
Switching between SBCS and DBCS and coexistence of SBCS and DBCS in the same session must be possible.

Example

The separation between single-byte and double-byte coded characters is currently implemented with SO/SI (shift out/shift in) control characters and other data stream dependent mechanisms. In the IBM 3270 data stream, separation between SBCS and DBCS is done by SO/SI, by field attribute, and by character attribute. In the IBM PC (which does code page switching), separation is done by using a predefined code point range to signify a double byte.

Keystroke Processing

Allowing the user to key in multiple SBCS characters (for example, Katakana or Hiragana) and, by a procedure, convert them into a single DBCS character (for example, Kanji) is necessary to support DBCS. This procedure is called "Kana to Kanji conversion" and is a technique by which the Japanese convert pronunciation to graphic form. For Chinese, it is necessary to allow components of a character to be combined and converted to a single character (for example, radical strokes to Chinese).

GUIDELINE 1300:

A method should be provided to form double-byte characters from multiple keystrokes.

Isolation of MRI at the System Resident Level

The implementation of double-byte character languages is a relatively recent activity. Although the product strategy is to migrate from the single-byte support to double-byte support, the user requirement to have both single-byte and double-byte support on a system will last a long time. Therefore, the MRI must be isolated at the system resident level to allow users to select, on a session basis, single-byte character or double-byte character language support.

RULE 630:
MRI modules for DBCS systems must be loaded separately from the executable code.

Appendix A. Checklist

The checklist that follows is intended for use in determining the rules and guidelines that have been followed by a product. When doing an assessment, determine whether the product follows the rule (YES), does not follow the rule (NO), or the rule does not apply (NA). Included also is the rule number (No.) and a page reference (Reference).

Base Subset Rules

YES	NO	NA	Rule	No.	Reference
☐	☐	☐	The existence of a specific character set within a system or its components must not be assumed.	010	Page 9
☐	☐	☐	Character sets must be selectable on request by the operator, a user, or an application.	020	Page 11
☐	☐	☐	Monocasing must be definable for each language and code page.	030	Page 12
☐	☐	☐	Folding must be definable for each language and code page.	040	Page 13
☐	☐	☐	Single-byte character set (SBCS) code points in the graphics ranges must be used only for graphic characters and must not be used for control purposes in hardware.	050	Page 14
☐	☐	☐	The use of a graphic character for software control purposes must not preclude the use of the same character in the text of messages, menus, prompts, or input or output fields.	060	Page 15
☐	☐	☐	The set of characters allowed for use in the entry of data must be definable by the system operator, a user, or an application.	070	Page 16
☐	☐	☐	Graphic symbols and icons must be universal or translatable.	080	Page 17
☐	☐	☐	Code points not currently assigned in standard code pages must not be assigned to new characters.	090	Page 17

YES	NO	NA	Rule	No.	Reference
☐	☐	☐	All characters on the active code page must be accessible.	100	Page 19
☐	☐	☐	Language dependent parts of a product must be isolated from nonlanguage dependent parts for easy modification.	110	Page 20
☐	☐	☐	The design of a product must allow for the national language support of the various components of the product to be independent of each other.	120	Page 21
☐	☐	☐	National language exits must be provided at strategic points.	130	Page 22
☐	☐	☐	Diagnostics must be enabled.	140	Page 22
☐	☐	☐	Character generators must produce at least 190 user-accessible graphic characters, excluding indicator characters and special icons.	150	Page 23
☐	☐	☐	Character generators must be built to be flexible so that the contents can be modified and expanded.	160	Page 25
☐	☐	☐	The brightness or density of a character element must be consistent for both horizontal and vertical lines.	170	Page 26
☐	☐	☐	For electronic font printers, a mode must be provided such that each successive pixel of a character can be printed.	180	Page 27
☐	☐	☐	Keyboard design must allow for software selection and control of its mode of operation.	190	Page 30
☐	☐	☐	Logical layouts different from a given physical keyboard layout must be available to the user.	200	Page 31
☐	☐	☐	If a product uses data or text processing keyboards, at least one of them must have 48 or more graphic keys.	210	Page 32
☐	☐	☐	The product design must allow the implementation of at least two graphic layers of three shift levels each.	220	Page 33
☐	☐	☐	The manufacturing process must not restrict the relative positioning of graphics within the valid area on the top or front face of the key.	230	Page 34
☐	☐	☐	Capslock must monocase to uppercase.	240	Page 34
☐	☐	☐	Each keyboard layer must have an independently implemented capslock and shiftlock function.	250	Page 35
☐	☐	☐	Any data key must be definable by the user to be nonescaping or repeating.	260	Page 36
☐	☐	☐	All MRI and presentation control information must be isolated from the executable code.	270	Page 39

YES	NO	NA	Rule	No.	Reference
☐	☐	☐	Sufficient space must be available for MRI expansion caused by translation.	280	Page 41
☐	☐	☐	Functions dependent on display field length and/or position must not be designed in such a way that they are affected by MRI expansion.	290	Page 42
☐	☐	☐	There must be a method provided to allow identification and tracking of panels and messages during the translation process.	300	Page 46
☐	☐	☐	Variables must be permitted to assume any location and order within a display field.	310	Page 47
☐	☐	☐	Messages and other displayed words or phrases must be complete entities and not be constructed from individual words or phrases.	320	Page 48
☐	☐	☐	End-user commands, keywords, and responses must be treated like other MRI.	330	Page 50
☐	☐	☐	Entry of end-user commands, keywords, or responses must be possible without regard to uppercase or lowercase.	340	Page 52
☐	☐	☐	A product with National Language Dependent Functions must be designed to facilitate addition of other countries or national languages.	350	Page 53
☐	☐	☐	Date and time formats must be selectable.	360	Page 55
☐	☐	☐	Numeric punctuation must be selectable.	370	Page 57
☐	☐	☐	Number rounding and mathematical formats must be selectable.	380	Page 58
☐	☐	☐	Monetary format must be definable.	390	Page 59
☐	☐	☐	The default currency symbol or its abbreviations must be selectable.	400	Page 60
☐	☐	☐	The currency symbol position must be selectable.	410	Page 60
☐	☐	☐	Field sizes for monetary values must be selectable.	420	Page 60
☐	☐	☐	The measurement system must be selectable.	430	Page 61

Bidirectional Subset Rules

YES	NO	NA	Rule	No.	Reference
☐	☐	☐	Hardware and system software design must allow for bidirectional data to be passed to applications in the same order a speaker of the language would spell it out.	440	Page 69
☐	☐	☐	The product design must allow for the implementation of the correct handling of bidirectional keyboard and presentation functions.	450	Page 70
☐	☐	☐	Designing of a function that implies logical movement of cursor or characters must permit mirroring of that function.	460	Page 74
☐	☐	☐	Keys or operations labeled with directional icons or symbols must perform according to the icon or symbol.	470	Page 74
☐	☐	☐	Keyboard nomenclature for mirrored functions must be independent of the direction of data or text entry.	480	Page 75
☐	☐	☐	An inverse move instruction must be part of each processor's instruction set.	490	Page 75
☐	☐	☐	Display functions must not assume a left-to-right orientation.	500	Page 76
☐	☐	☐	Field attributes must contain room for directional information.	510	Page 76
☐	☐	☐	Indicator location must be reserved for the current direction of the cursor (direction of input).	520	Page 77
☐	☐	☐	The design must allow for independent handling of graphic and text orientation.	530	Page 77
☐	☐	☐	Provision must be made to allow shape determination to be performed.	540	Page 79
☐	☐	☐	The deshaping must be definable.	550	Page 80
☐	☐	☐	Provision must be made to allow the selection of the appropriate presentation shape for the numerals.	560	Page 81
☐	☐	☐	Characters must be allowed to touch each other on printers and displays.	570	Page 81

Double-Byte Character Subset Rules

YES	NO	NA	Rule	No.	Reference
☐	☐	☐	DBCS code points in the graphic character range must be used only for graphic characters and must not be used for control purposes.	580	Page 85
☐	☐	☐	Single byte meaning must not be drawn from either byte of double-byte data.	590	Page 86
☐	☐	☐	DBCS character generators must be capable of producing at least 36 101 user-accessible graphic characters, excluding indicator characters and special icons.	600	Page 86
☐	☐	☐	Additional DBCS characters must be definable by the operator, a user, or an application.	610	Page 87
☐	☐	☐	Switching between SBCS and DBCS and coexistence of SBCS and DBCS in the same session must be possible.	620	Page 87
☐	☐	☐	MRI modules for DBCS systems must be loaded separately from the executable code.	630	Page 88

Base Subset Guidelines

YES	NO	NA	Guideline	No.	Reference
☐	☐	☐	Lowercase alphabets should not be assumed to be invariant.	1010	Page 10
☐	☐	☐	Character sets should be definable by the operator, a user, or an application.	1020	Page 12
☐	☐	☐	Special characters, including punctuation marks, should be definable and not program dependent.	1030	Page 17
☐	☐	☐	The characters to be underscored should be selectable.	1040	Page 18
☐	☐	☐	Controllers should support concurrent handling of devices requiring multiple or different code pages.	1050	Page 20
☐	☐	☐	Microcode and software should be modular and designed to be easily modified.	1060	Page 21
☐	☐	☐	The matrix of pixels forming a character cell should be not less than 9 pixels wide and not less than 16 pixels high.	1070	Page 24
☐	☐	☐	For hard font printers, the capability of printing at least 190 characters without a print element change should be provided.	1080	Page 26
☐	☐	☐	Logical keyboard layouts should be user definable.	1090	Page 32
☐	☐	☐	Indicators should be provided to show the level and the layer a keyboard is operating in.	1100	Page 33
☐	☐	☐	A keyboard should be capable of identifying its physical layout to the keystroke processing logic.	1110	Page 36
☐	☐	☐	MRI modules should be packaged separately from the executable code.	1120	Page 40
☐	☐	☐	MRI modules for SBCS systems should be loaded separately from the executable code.	1130	Page 41
☐	☐	☐	A consistent convention should be used throughout the product for denoting variables and input fields.	1140	Page 46
☐	☐	☐	Words should not be used in place of numbers.	1150	Page 59
☐	☐	☐	The terminology used in MRI should be consistent throughout a product.	1160	Page 63
☐	☐	☐	Abbreviations should be avoided.	1170	Page 63
☐	☐	☐	Slang, jargon, and humor should not be used.	1180	Page 64

YES	NO	NA	Guideline	No.	Reference
☐	☐	☐	Trademarks should be identified and explained.	1190	Page 64
☐	☐	☐	Ambiguous words should not be used.	1200	Page 65
☐	☐	☐	Proper style and sentence structure should be used in MRI.	1210	Page 65
☐	☐	☐	Negative questions should be avoided.	1220	Page 66

Bidirectional Subset Guidelines

YES	NO	NA	Guideline	No.	Reference
☐	☐	☐	Indicator locations should be reserved for screen and field orientation, current level of nesting, status of push (nesting mechanism), and status of symmetric swapping.	1230	Page 77
☐	☐	☐	The design should provide for a method to indicate to the user the nesting structure of a string.	1240	Page 77
☐	☐	☐	A system wide method of deshaping Arabic characters or character strings should be provided.	1250	Page 80
☐	☐	☐	An indicator location should be provided for the status of shape determination.	1260	Page 80
☐	☐	☐	Printers should be designed so that proportional spacing can be provided.	1270	Page 82
☐	☐	☐	A method should be provided to allow alignment of the baseline of Arabic and Latin characters (including "Arabic" numerals).	1280	Page 82

Double-Byte Character Subset Guidelines

YES	NO	NA	Guideline	No.	Reference
☐	☐	☐	For DBCS, the matrix of pixels forming a character should not be less than 24 pixels wide and not less than 24 pixels high.	1290	Page 87
☐	☐	☐	A method should be provided to form double-byte characters from multiple keystrokes.	1300	Page 88

Appendix B. Country Groupings by Language Characteristics

The countries in the following list are covered in Volumes 2, 3, and 4 of this guide.

- Countries with left-to-right languages using single-byte coded characters
- Countries with a bidirectional language using single-byte coded characters
- Countries that use double-byte coded characters.

Note: For further information on the languages used by the countries in these lists, consult volumes 2, 3, and 4 of this guide or Appendix C, "National Language Standards and Laws."

Group 1: Left-To-Right Single-Byte Language Countries

Albania
Argentina
Austria
Belgium
Bulgaria
Canada
Czechoslovakia
Denmark
Finland
France
German Democratic Republic
Germany, Federal Republic of
Greece
Hungary
Iceland
Italy
Japan (Kana or Latin)
Korea, Republic of
Netherlands
New Zealand
Norway
People's Republic of China (Latin)
Poland
Portugal
Romania
South Africa
Spain
Sweden
Switzerland
Thailand
Turkey
United Kingdom
United States of America
Yugoslavia

Group 2: Bidirectional Single-Byte Language Countries

Algeria
Bahrain
Egypt
Iran
Iraq
Israel
Jordan
Kuwait
Lebanon
Libya
Morocco
North Yemen
Oman
Pakistan
Qatar
Saudi Arabia
South Yemen
Sudan
Syria
Tunisia
United Arab Emirates

Group 3: Double-Byte Language Countries

Japan
Korea
People's Republic of China
Republic of China

Appendix C. National Language Standards and Laws

This appendix lists:

- Some standards organizations and their addresses
- Some language standards that relate to graphic symbols
- Some language laws that have been passed by various countries.

The following lists of standards organizations, standards, and national laws are not an exhaustive compilation. The standards organizations and national governments should be consulted for current and complete versions of the material.

Language-Related Standards

Standards Organizations' Addresses

AFNOR	Association française de normalisation Tour Europe, Cedex 7, 92080 PARIS LA DEFENSE
ANSI	American National Standards Institute 1430 Broadway, New York, N.Y. 10018
ASMO	Arab Standards and Metrology Organization P.O. Box 926161, Amman, Jordan
CAS	China Association for Standards P.O. Box 820, Beijing, China
CCITT	Consultative Committee for International Telegraph and Telephone 2, rue de Varembe, CH-1211 Geneva 20, Switzerland
CSA	Canadian Standards Association 178 Rexdale Bvld, Rexdale, Ontario, M9W 1R3
DIN	Deutsches Institut fuer Normung Burggrafenstrasse 4-10, Postfach 1107, D-1000 Berlin 30
ECMA	European Computer Manufacturers Association 114 Rue du Rhone, 1204 Geneva, Switzerland

FIPS	Federal Information Processing Standards U.S. National Bureau of Standards, Institute for Computer Sciences and Technology, Gaithersburg, MD 20899
IEC	International Electrotechnical Commission 3, rue de Verambe, CH-1211 Geneva 20, Switzerland
ISO	International Standards Organization 1, rue de Verambe, Case Postale 56, CH-1211 Geneva 20, Switzerland
JIS	Japanese Industrial Standards Committee c/o Standards Department, Agency of Industrial Science and Technology Ministry of International Trade and Industry, 1-3-1, Kasumigaseki Chiyoda-ku, Tokyo 100
NNI	Nederlands Normalisatie-instituut Kalfjeslaan, P.O. Box 5059, 2600 GB Delft
SFS	Suomen Standardisoimisliitto (Finland) P.O. Box 205, SF-00121, Helsinki
SCC	Standards Council of Canada International Standardizations Branch, 2000 Argentina Road, Suite 2-401, Missassauga, Ontario, L5N 1V8

Sample National Language Standards

AFNOR NF 62-10	French 7-Bit Code
ANSI BSR X3.134.1	8-Bit ASCII Structure and Rules
ANSI BSR X3.134.2	8-Bit ASCII Supplemental Multilingual Graphic Character Set
ANSI X3.110-1983--CSA T500	Videotex/Teletex Presentation Level Protocol Syntax - North American PLPS
ANSI X3.32	Graphic Representations of the Control Characters of ASCII
ANSI X3.4	American National Standard Code for Information Interchange
ANSI X3.41	Code Extension Techniques for Use with ASCII
ANSI X3.64	Additional Controls for Use with ASCII
ANSI X4.16	American National Standard Magnetic Stripe Encoding for Credit Cards
ANSI X4.22	For Office Machines and Supplies - Alphanumeric Machines - Alternate Keyboard Arrangement
ANSI X4.23	For Office Machines and Supplies - Alphanumeric Machines - Keyboard Arrangement

ASMO 445	Bilingual Arabic Latin telex 5-bit code and keyboard
ASMO 449	7-Bit coded Arabic character set for information interchange
ASMO 584	Conversions between ASMO 445 and ASMO 449
ASMO 662	8-Bit coded Arabic character set for information interchange
ASMO 663	Arabic keyboard terminal layout
ASMO 708	8-Bit coded Arabic/English character set for information interchange
CAS GB1988-80	7-Bit Coded Character Sets for Information Processing and Interchange
CAS GB2311-80	Information Processing - 7-Bit Coded Character Set - Code Extension Techniques
CAS GB2312-80	Code of Chinese Graphic Character Set for Information Interchange - Primary Set
CCITT T.61	Character Repertoire and Coded Character Sets for the International Teletex Service
CCITT T.100.1984	International Information Exchange for Interactive Videotex
CSA Z243.4-1985	7-Bit and 8-Bit Coded Character Sets for Information Processing and Interchange
DIN 2137	German Standard: Office Machines, Alphanumeric Arrangement
DIN 66003	Informationsverarbeitung 7-Bit Code (German 7 Bit Code)
FIPS PUB 15	Federal Information Processing Standards - Subsets of the Standard Code for Information Interchange
IEC - 417	Graphic Symbols for Use on Equipment
ISO DIS 4882	Office Machines and Data Processing Equipment, Line Spacings, and Character Spacings
ISO DIS 6329	Symbols for Duplicating and Document Copying Machines
ISO DIS 8613	Office Document Architecture
ISO-IR	International Register of Coded Character Sets to be Used with Escape Sequences - Registration Authority: ECMA, Geneva
ISO R1090	Functions Key Symbols for Typewriters
ISO R1093	Keytop and Printed or Displayed Symbols for Adding Machines and Calculating Machines

ISO 646	Information Processing - 7-Bit Coded Character Set for Information Interchange
ISO 1091	Typewriters - Layout of Printing and Function Keys
ISO 2022	Information Processing - ISO 7-Bit and 8-Bit Coded Character Sets - Code Extension Techniques
ISO 2126	Basic Arrangement for the Alphanumeric Section of Keyboards Operated with Both Hands
ISO 2375	Data Processing - Procedure for Registration of Escape Sequences
ISO 2530	Keyboard for international information processing interchange using the ISO 7-bit coded character set - Alphanumeric area
ISO 3243	Keyboards for Countries Whose Languages Have Alphabetic Extenders - Guidelines for Harmonization
ISO 3461	Graphic Symbols, General Principles for Presentation
ISO 4062	Dictation Equipment Symbols
ISO 4197	Office Machines - Keyboards - Key Numbering System and Layout Charts
ISO 4217	Codes for the representation of currency and funds
ISO 4873	Information Processing - 8 Bit Code for Information Interchange - Structure and Rules for Implementation
ISO 6429	Information Processing - ISO 7-Bit and 8-Bit Coded Character Sets - Additional Control Functions for Character Imaging Devices
ISO 6937	Information Processing - Coded Character Sets for Text Communication
ISO 7350	Text Communication - Registration of Graphic Character Sub-repertoires
ISO 8859	8-Bit Single-Byte Coded Graphic Character Sets
ISO 8884	Common Secondary Keyboard Layout for Multiple Latin-Alphabet Languages
JIS C 6226	Code of the Japanese Graphic Character Set for Information Interchange
JIS C 6233	Keyboard Layout for Information Processing Using the JIS 7-bit Coded Character Set
NNI NEN 2294	Netherlands Standard Keyboard Layout
SFS 3548	Alphanumeric Keyboard

National Laws

Most countries have language laws affecting the importation, sale, or use of data processing equipment, software, or documentation. Some of the laws specify the language(s) to be used on labels, keyboards, documentation, or software. Other laws regulate cultural aspects such as date formats, calendars, or numeric representation. Because there are many laws and constant revision of laws, only the country issuing a law can adequately describe it. The following are a few examples of language laws that have been issued (but may not be current):

Canada: Quebec Law 101, Charter of the French Language, August 26, 1977; Canadian Official Language Act of 1969.

Sweden: The Work Environment Act, 1st July 1978.

Venezuela: Consumer Protection Law, Article 10.

Glossary of Terms

This glossary includes definitions of some terms found in this document. For the definition of terms not found in this glossary, refer to IBM publication *Dictionary of Computing,* GC20-1699.

Some of the terms defined below are from *The American National Dictionary for Information Processing,* copyright 1977 by the Computer and Business Equipment Manufacturers Association, copies of which may be purchased from the American National Standards Institute at 1430 Broadway, New York, New York 10018. These definitions are identified by an asterisk (*).

The symbol (ISO) at the beginning of a definition indicates that it is reproduced from a published section of the International Organization for Standardization *Vocabulary of Data Processing* or from a published section of the ISO *Vocabulary of Office Machines.*

A

active code page. The code page to which a device is logically connected at a particular point in time.

active keyboard. The character set layout that is available for data or text entry and that is, at most, the set of characters on the active code page.

AFNOR. Association française de normalisation.

algorithm*. A finite set of well-defined rules for the solution of a problem in a finite number of steps. For example, a full statement of an arithmetic procedure for evaluating sin x to a stated precision.

ANSI. American National Standards Institute.

application. The use to which a data processing system is put; for example, a payroll application, an airline reservation application, a network application.

Arabic shapes for numerals. The shapes 1, 2, 3, 4, 5, 6, 7, 8, 9, and 0. See also Hindi shapes for numerals.

ASCII*. American Standard Code for Information Interchange. The standard code, using a coded set consisting of 7-bit coded characters (8 bits including parity check), used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

ASMO. Arab Standards and Metrology Organization.

B

base shape. The shape of a character such as an Arabic character, which identifies the character but not its presentation shape. See also presentation shape.

bidirectional languages. Languages using text oriented from right to left, with numeric and other text oriented from left to right.

C

CAS. China Association for Standards.

CCITT. Consultative Committee for International Telephone and Telegraph.

CEPT. Commission of European Post and Telegraph.

character cell. The maximum physical boundary of a single character. For example, on the 3800 Printing Subsystem, a cell is made up of 24 rows with total height of 4.23 mm and 18 bit positions having a total width of 2.54 mm.

character generator. (1) In computer graphics, a functional unit that converts the coded representation of a graphic character into the shape of the character for display or print. (2) In word processing, the means for generating visual characters or symbols for coded data.

character set. A defined set of characters. No coded representation is assumed.

code page. A specification of code points for each graphic character in a set or in a collection of graphic character sets. Within a code page, a code point can have one and only one specific meaning.

code point. The specific binary value used to encode a specific character on a code page.

coded character set*. (ISO) A set of unambiguous rules that establish a character set and the one-to-one relationships between the characters of the set and their coded representations.

command. End-user command (MRI).

component. A hardware or software entity forming part of a system. A piece of logic that controls the operation of a device.

CSA. Canadian Standards Association.

CSU. Customer setup. The unpacking, setup, and checkout of IBM CSU-designated machines by user personnel, according to a sequence of instructions provided by IBM, without the use of tools or the assistance of IBM personnel.

CUA. Common User Access. A basic element of Systems Application Architecture (SAA).

cursive script. Script that uses characters connected to each other.

D

data base*. A collection of data fundamental to a system.

data entry. The method of entering data into a computer system for processing, usually in a field-oriented environment where the entry is governed by a program. See text entry for comparison.

data key. A key supporting a graphic character.

data stream. A continuous stream of data elements being transmitted, or intended for transmission, in character or binary-digit form, using a defined format.

DBCS. See double-byte character set.

dead key. On a typewriter and in word processing, a key that allows a character to be typed without the imprint position being changed. This term should be replaced with *nonescaping key*.

DIN. Deutsches Institut fuer Normung.

display field. An area on a display space that contains a set of characters that can be manipulated or operated upon as a unit (for example, prompts, messages).

dot matrix. (1) In computer graphics, a two-dimensional pattern of dots used for constructing a display image. This type of matrix can be used to represent characters by dots. (2) In word processing, a pattern of dots used to form characters. This term normally refers to a small section of the set of addressable points; for example, a representation of characters by dots.

double-byte character set. A character set in which each character is represented by a 2-byte code. Some character sets, such as Kanji, which is used in Japan, are too rich in symbols to be able to represent all the characters using 1-byte codes. A double-byte character set is used to represent the symbols that make up these larger character sets.

E

EBCDIC. Extended Binary Coded Decimal Interchange Code. A coded character set consisting of 8-bit coded characters.

electronic font printer. A printer using a technique in which the character image is formed by a matrix of pixels.

ECMA. European Computer Manufacturers Association.

enable (national languages). To design a product in such a way as to facilitate the inclusion of national language functions.

end user. A person, process, program, device, or system that employs a user application network for the purpose of data processing and information exchange.

EPROM. Erasable programmable read-only memory.

EZ-VU. A set of programs for the IBM Personal Computer that help users design and use screens for interactive programs.

F

field attributes. The data description governing the presentation and handling of data in the associated data field.

FIPS. Federal Information Processing Standards.

folding. The substitution of one character for another, generally to map a larger character set into a subset. Folding may result in the loss of information. For example, folding allows the printing of uppercase graphic characters when lowercase characters are not available in the character array on the printer chain or train.

font*. A family or assortment of characters of a given size and style; for example, 9 point Bodoni Modern.

G

graphic character*. (ISO) A character, other than a control character, that is normally represented by a graphic.

graphic character set. A defined set of graphic characters. No coded representation is assumed.

H

hard-coded. A value coded in a program as a constant as opposed to a variable.

hard font printer. A printer that uses a print technique in which the characters are physically engraved on the print element.

Hindi shapes for numerals. A set of shapes used in many Arabic countries instead of the more familiar "Arabic" ones. See also Arabic shapes for numerals.

Hiragana. A set of phonetic symbols (syllabary) used in Japanese in conjunction with Kanji and other character sets. See also Katakana.

I

icon. A graphic character used to represent the presence or meaning of a function symbolically.

ideographic language. A written language in which each character represents a thing or an idea. An example of such a language is Chinese. See also phonetic language.

IEC. International Electrotechnical Commission.

implement (national languages). To develop, integrate, test, and release specific national language functions on a product.

indicator symbol or light. A symbol or light that shows the status of a function.

instruction. A programming language instruction that produces executable code (not MRI).

invariant character set. (1) A character set, such as the Syntactic Character Set, that does not change from code page to code page. (2) A minimum set of characters that is available as part of all character sets.

inverse move. An instruction that moves one field in storage to another, reversing the order of bytes as it proceeds.

ISO. International Organization for Standardization.

ISO environment. Structure defined by ISO 2022.

J

JIS. Japanese Industrial Standards.

K

Kanji. A character set of symbols used in Japanese ideographic script.

Katakana. Katakana is a set of phonetic symbols (syllabary) used in Japanese. It is often used to express foreign words phonetically. See also Hiragana.

L

language*. (ISO) A set of characters, conventions, and rules that is used for conveying information. The three aspects of language are pragmatics, semantics, and syntax.

language shift. In a multishift keyboard, a shift that takes the user from one language to another. For example, the language shift is used to switch between Hebrew and English when mixed text is entered.

Latin alphabet. An alphabet comprising the letters a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, and z in uppercase and lowercase, with or without accents.

Latin #1 character set. The term used in IBM product literature to designate the character set for most of Western Europe and North America.

LF. Linguistic function. Examples include spell checking, hyphenation, and dictionary functions.

lowercase. The small letters a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, and z, and other characters in the lower shift. See also uppercase.

M

mirror function. Functions, such as Tab and Carriage Return, that should work in a mirror-image manner depending on the current orientation in a bidirectional environment.

mnemonic symbol*. (ISO) A symbol chosen to assist the human memory. For example, an abbreviation such as *mpy* for *multiply.*

monocasing. The translation of alphabetic characters from one case (usually the lowercase) to their equivalents in another case (usually the uppercase).

MRI. Machine readable information. All textual information contained in a program such as a system control program, an application program, or microcode. MRI includes all information that is presented to or received from a user interacting with a system. This includes menus, prompts, messages, report headings, commands, and responses. MRI may appear on printers or on display panels. Contrast with machine readable material (MRM), which includes MRI, executable code, and constants.

N

national language support. The ability for a user to communicate with products in a language other than English used in the U.S.A.

national use graphics. Graphics on a code page not forming part of the invariant character set.

NLDF. National Language Dependent Function. Any function of a product that must be altered to suit a new language or country. Examples include sort and calendar functions.

NLS. National language support.

NNI. Nederlands Normalisatie-instituut.

nomenclature. A set of terms and/or symbols such as those that appear on keys, switches, and labels.

nonescaping key. A key that allows a character to be typed without the presentation position being changed.

non-Latin-based alphabet. An alphabet comprising letters other than the Latin based ones. Such alphabets as used in Greek and Arabic are typical.

O

operator. The system operator.

P

panel. Information appearing on a display screen.

pel. Picture element. Also called a pixel.

phonetic language. A written language in which each character represents a sound. Examples of phonetic languages are English, Greek, and Russian. See also ideographic language.

PRC. Peoples Republic of China.

picture element. The smallest element of a display space or print space that can be individually addressed.

pixel. Picture element. Also called a pel.

presentation. Printing or displaying.

presentation shape. The shape of a character such as an Arabic character that is to be presented to the user. See also base shape and shape determination.

product. Hardware or software or any combination marketed as an individual unit, separately orderable by a customer.

PROM. Programmable read-only memory.

Q

quadrant. One of four groups of characters, each group being represented by a combination of up/down shift and right/left shift in a four-shift keyboard where each key may represent as many as four characters, one from each quadrant.

R

RAM. Random access memory.

repeat key. A key that repeats its function when pressed and held down.

retrofit (national languages). To implement national language function on an unenabled product.

right-to-left mode. An input mode of a bidirectional machine in which the cursor moves to the left after the entry of each character.

ROM. (ISO) Read only memory.

S

SBCS. Single-byte character set.

SCC. Standards Council of Canada.

session. The period of time between logon and logoff without re-ipl.

SFS. Suomen Standardisoimisliitto (Finland).

shadow keyboard. A keyboard layout that is different from that engraved on the actual keyboard and that may be used to access characters not available through the engraved keyboard.

shape determination. A process that decides which of the several (up to four) shapes an Arabic character is to be used in current context. For each character, the decision is based on the linking capabilities of current and surrounding characters.

single-byte character set. A character set in which each character is represented by a one-byte code.

special character. A graphic character that is not a letter, not a digit, and not a space character.

system. A collection of interconnected hardware and software organized to accomplish a set of specific functions.

T

text entry. Entry of characters in a free format. See data entry for comparison

U

uppercase. The capital letters A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, and Z, and other characters in the upper shift. See also lowercase.

user. See end user.

W

window. An area of the screen through which a panel of information is displayed.

Index

National Language Information
and Design Guide
Volume 1
Designing Enabled Products,
Rules and Guidelines

SE09-8001-00

READER'S COMMENT FORM

Please use this form only to identify publication errors or to request changes in publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your nearest IBM branch office. You may use this form to communicate your comments about this publication, its organization, or subject matter **with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.**

☐ If your comment does not need a reply (for example, pointing out a typing error), check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

☐ If you would like a reply, check this box. Be sure to print your name and address below.

Page number(s): Comment(s):

Please contact your nearest IBM branch office to request additional publications.

Name ______________________________

Company or
Organization ______________________________

Address ______________________________

____

SE09-8001-00

Reader's Comment Form

CUT
OR
FOLD
ALONG
LINE

Fold and tape | Please Do Not Staple | Fold and tape

IBM Canada Ltd. Laboratory
National Language Technical Centre
Dept. 979
1150 Eglinton Avenue East
North York, Ontario, Canada
M3C 1H7

Fold and tape | Please Do Not Staple | Fold and tape

IBM ®

National Language Information and Design Guide
Volume 1
Designing Enabled Products, Rules and Guidelines

SE09-8001-00

READER'S COMMENT FORM

Please use this form only to identify publication errors or to request changes in publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your nearest IBM branch office. You may use this form to communicate your comments about this publication, its organization, or subject matter **with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.**

☐ If your comment does not need a reply (for example, pointing out a typing error), check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

☐ If you would like a reply, check this box. Be sure to print your name and address below.

Page number(s): Comment(s):

Please contact your nearest IBM branch office to request additional publications.

Name ____________________

Company or Organization ____________________

Address ____________________

____

SE09-8001-00

Reader's Comment Form

CUT
OR
FOLD
ALONG
LINE

Fold and tape — **Please Do Not Staple** — Fold and tape

IBM Canada Ltd. Laboratory
National Language Technical Centre
Dept. 979
1150 Eglinton Avenue East
North York, Ontario, Canada
M3C 1H7

Fold and tape — **Please Do Not Staple** — Fold and tape

IBM®

International Business Machines Corporation

SE09-8001-00

Order Number SE09-8001-00

Printed in U.S.A.