Reuter Multilingual Text Encoding Standard

Beta site version

Table of contents

0 Change history

Version	***Date***	***Comments***
0.20	3 Sep 1993	Beta site version
0.30	15 Jul 1994	Beta site version 1 A new appendix has been added, restricting the use of some mechanisms. 2 Minor errors have been corrected.

1 Introduction

1.1 Purpose

The Reuter Multilingual Text Encoding Standard defines a set of rules governing the interchange of multilingual textual information between a producing application and a consuming application.

The Standard seeks to give application writers all the information they need in order to display or otherwise process multilingual text delivered electronically by Reuters.

1.2 Scope

This Standard defines the encoding of all textual fields generated by Reuters. It does not, however, define which fields are textual and which are not. Furthermore, specific Reuter products may support – whether in general, or in certain textual fields – only some of the character sets specified by this Standard. Similarly, the various Reuter products have their own rules whereby the beginnings and ends of textual fields are identified, and these rules are outside the scope of this Standard. These matters are all addressed within product-specific documentation.

1.3 Background

Reuters generates textual information in a varied and growing number of languages. Some of these languages use identical character sets, others differ from one another in just a few characters, and yet others use completely different character sets.

The Reuter Multilingual Text Encoding Standard defines the representation of these characters within computing and communications systems. This Standard is based on the International Standard ISO 2022, titled *Information Processing – ISO 7-bit and 8-bit coded character sets – Code extension techniques*.

ISO 2022 defines the rules which govern the handling of character sets, but does not define the character sets themselves. Instead, it refers to the *International Register of Coded Character Sets to be used with Escape Sequences*, together with an *International Registration Authority* tasked with maintaining this Register. *ECMA*, the European Computer Manufacturers Association, has been designated as the International Registration Authority.

The International Standard ISO 2022 does not demand full conformance. It permits substantial discretion in the implementation of subsets and of user-defined features. Reuters has made use of this discretion in defining the Reuter Multilingual Text Encoding Standard.

This document, in general, makes no distinction between those requirements which form part of ISO 2022 and those requirements which result from subsequent decisions on the part of Reuters. In the case of any discrepancies between ISO 2022 and this Reuter Standard, this Standard takes precedence.

The reader of this document should not need to refer to ISO 2022 in order to implement an application which receives Reuter Multilingual Text.

Though the primary subject matter of this document is the encoding of graphic characters, both ISO 2022 and this Reuter Standard deal also with the encoding of control functions.

1.4 Structure

This Standard is structured in a manner which separates the description of the Reuter Multilingual Text Encoding Scheme from a description of the particular character sets and function sets supported by the Standard. This separation should assist application writers in designing applications which are robust and are not adversely affected by any future increase in the number of character sets or function sets supported.

Chapter 2, which defines the Reuter Multilingual Text Encoding Scheme, is intended to be read from start to finish rather than to be treated as a reference text. Many of the concepts described are highly inter-related. They have been linked in a manner which should make it possible for the reader to digest one concept before moving on to another. An individual section taken in isolation, without the sections which precede it, may not be meaningful to the reader.

1.5 Restrictions on producing applications

As some existing consuming applications cannot handle the full range of mechanisms specified by this Standard, certain restrictions (described in appendix H) have been placed upon producing applications. Some of these restrictions may be lifted at a future time. It follows that designers of consuming applications should disregard these restrictions, so that their applications continue to operate if and when the restrictions no longer apply.

1.6 Terminology

A number of terms used within this Standard have a precise technical meaning which does not exactly match their meaning in day-to-day English. All technical terms are italicised at the point of their first occurrence and are also included in the Glossary, to be found in appendix A.

ISO 2022 terminology has been adhered to where it contributed to the clarity of the text. In those places where the International Standard was felt to be obscure, alternative terms have been chosen.

1.7 Supported character sets and function sets

As well as defining the encoding scheme itself, this Standard defines the application of the scheme to a number of specific character sets and function sets. The character and function sets currently supported by this Standard are described in appendices to this document.

From time to time further character sets and function sets may be included in this Standard. Although no guarantee can be given, the intention is that such changes should have no impact on the handling of currently supported character and function sets.

Though many of the character and function sets supported by this Standard are included in the International Register, others have been defined by Reuters for private use and have not been registered with ECMA. Furthermore, the treatment of certain character and function sets as defined by this Standard differs in some respects from that implied by their entries in the International Register. Full details of such differences are provided within this document.

1.8 Problems

Any errors or ambiguities encountered in this Standard should be brought to the attention of the local Customer Support Group.

2 Multilingual Text Encoding Scheme

2.1 Introduction

The Reuter Multilingual Text Encoding Standard defines a set of rules governing the interchange of multilingual textual information between a producing application and a consuming application.

2.2 Tokens

The multilingual textual information is made up of *tokens*. A given token may be:

- a *graphic character*, often abbreviated to *character*,
- a *control function*, or
- a *mapping function*.

2.3 Bytes

A *byte* is an ordered set of eight bits, treated as a unit. Bytes are used in the *representation* of tokens.

2.4 Context

Each byte has only 256 possible *values*. This Standard permits the representation of a much larger number of distinct tokens. This increase in capacity is accomplished by the use of a *current context* and of *byte sequences*.

At the *beginning of an information interchange* the current context is set to the *initial context*. The current context is changed during an information interchange by the use of *mapping functions*. Changes to the current context are cumulative.

2.5 Byte sequences

Each token is represented by a *byte sequence*. The byte sequences defined by this Standard vary in length from a single byte to many bytes.

Though the values of the bytes within a given byte sequence can be established by examination of that byte sequence in isolation, the *meaning* of the byte sequence depends on the context within which it is encountered. The same byte sequence encountered within two different contexts represents two different tokens.

A byte sequence which represents some token within one context may not represent any token at all within some other context. A byte sequence encountered within a context where it does not represent a token is an illegal byte sequence.

During an information interchange, a succession of tokens is passed from a producing application to a consuming application. Each token is represented by a complete byte sequence. The byte sequence representing a token cannot commence until the byte sequence representing the previous token has completed.

2.6 Byte stream

The succession of tokens passed from producing application to consuming application is represented by a succession of byte sequences. The ordered set of bytes making up all of these byte sequences is the *byte stream*.

Owing to the cumulative effect of mapping functions, the byte stream is meaningful only when read in the forward direction, starting from the beginning of an information interchange.

2.7 Functions

A function is either:

- a *mapping function*, or
- a *control function*.

A mapping function is a function defined by this Standard as a mechanism for changing the current context.

A control function is any function which is not a mapping function. An example of a control function is CARRIAGE RETURN.

2.8 Notation

The 256 distinct byte values are the building blocks used to construct legal byte sequences, which in turn represent legal tokens. The 256 byte values are shown graphically as a 16 x 16 *matrix* (see figure 2.1). Each of the 256 *cells* within the matrix is equivalent to one of the 256 possible byte values.

Two notations are used within this Standard:

- a hexadecimal notation, familiar to software designers dealing with byte values, and
- a decimal notation, used in many character set standards.

In both cases, a given cell is identified by its column number and row number within the matrix. The outer column and row numbers shown in figure 2.1 are decimal, the inner (italicised) column and row numbers are hexadecimal.

The decimal notation for the lower grey cell in figure 2.1 is 07/15, ie the decimal column number, followed by the character "/", followed by the decimal row number. Both the column number and the row number are always written using two decimal digits. The decimal notation for the upper grey cell is 02/00.

The hexadecimal notation for the lower grey cell is 7F, ie the hexadecimal column number, followed by the hexadecimal row number. Both the column number and the row number are always written using one hexadecimal digit. The hexadecimal notation for the upper grey cell is 20.

Wherever a byte value is given within this document, if the *nn/nn* format is *not* used, then the value should be interpreted as hexadecimal. Numbers other than byte values use ordinary decimal notation. The context makes clear whether a number is a byte value or not. For instance, the number "80" in the sentence:

> "The value of a byte with all bits except the top bit set to ZERO, and the top bit alone set to ONE, is 80."

should be interpreted as hexadecimal. On the other hand, the number "256" in the sentence:

> "Most of the 256 possible byte values have no fixed meaning."

is written using ordinary decimal notation.

A cell's co-ordinates, when expressed in hexadecimal form, are identical to the value of the byte to which the cell is equivalent. For example, the cell whose co-ordinates are written as 7F is equivalent to the byte whose value is 7F. This permits us to treat the two concepts (the cell co-ordinates and the corresponding byte value) as being interchangeable.

2.9 Areas

As well as being divided into 256 cells, the matrix shown in figure 2.1 is divided into four *areas*. Two of the areas, named CL and CR, contain 32 cells each. The other two areas, named GL and GR, contain 96 cells each.

The letters used to construct the names of the four areas have the following meanings:

- C stands for Control function,
- L stands for Left,
- R stands for Right, and
- G stands for Graphic character.

Areas CL and CR are used for the representation of control functions. Each byte sequence representing a control function consists of a single byte.

Areas GL and GR are used for the representation of graphic characters. A byte sequence representing a character may contain one byte or more than one byte. All the bytes within a byte sequence representing a particular character must fall within the same area.

2.10 Escape

In general, the cells of the matrix do not have permanently assigned meanings. Their meanings depend upon the current context, and change through the use of mapping functions.

There are a few exceptions to this rule. One of these exceptions is ESCAPE, abbreviated to ESC, which is always represented by a single byte in area CL with the value 1B (01/11).

2.11 Escape sequences

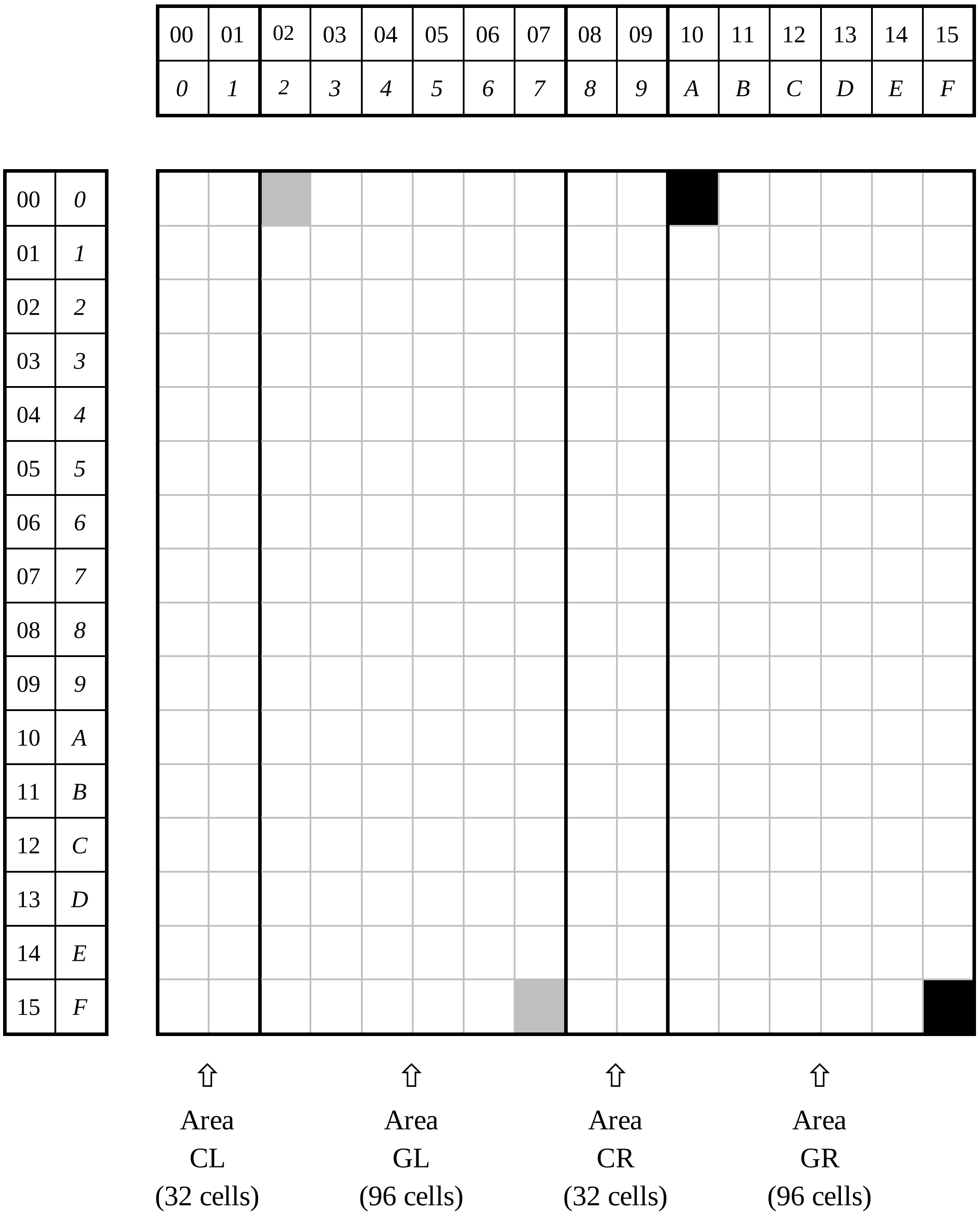
In most cases all the bytes within a byte sequence fall within a single area of the matrix. The sole exception is formed by *escape sequences*. Some (though not all) mapping functions are represented by these special byte sequences.

An escape sequence is a byte sequence which spans the two areas CL and GL. The byte sequence:

- starts with ESC, and
- contains one or more additional bytes drawn from area GL.

The only GL byte value not permitted in an escape sequence is 7F (07/15).

The meaning of an escape sequence is independent of the current context, and depends only on the values of the additional bytes.



The outer column and row numbers are decimal, the inner (italicised) numbers are hexadecimal.

Figure 2.1 – Matrix of cells, showing the four areas

2.12 Token sets

Both graphic characters and control functions are organised into sets of related tokens. Such a *token set* is either:

- a *character set* (ie a set of graphic characters), or
- a *function set* (ie a set of control functions).

Each token in a set is represented, within an appropriate context, by some byte sequence.

Unlike graphic characters and control functions, mapping functions are not organised into token sets.

2.13 Mapping

Token sets may be *mapped* onto various of the four areas. When a token set is mapped onto an area, byte values within that area are used to construct byte sequences representing tokens in that token set.

Function sets fall into two groups:

- those associated with area CL, and
- those associated with area CR.

A function set can be mapped only onto the area with which it is associated.

The same restriction does not apply to character sets. A character set may at one time be mapped onto area GL, and at another time onto area GR.

2.14 Top bit

The *top bit* is the most significant bit of a byte. The value of a byte with all bits except the top bit set to ZERO, and the top bit alone set to ONE, is 80.

1	0	0	0	0	0	0	0

Figure 2.2 – Top bit alone set to ONE

Examining the four areas (see figure 2.1) we see that:

- byte values in area CL run from 00 to 1F,
- byte values in area GL run from 20 to 7F,
- byte values in area CR run from 80 to 9F, and
- byte values in area GR run from A0 to FF.

Hence, byte values within areas CL and GL all have their top bit set to ZERO, while byte values within areas CR and GR all have their top bit set to ONE.

When a particular character set is mapped onto area GL, the bytes representing characters in that set will all have their top bit set to ZERO. When that same character set is mapped onto area GR, the bytes representing characters in that set will all have their top bit set to ONE.

The top bit, when combined with the current context, serves to determine which character set is being represented. The remaining seven bits of each byte within the byte sequence determine which particular character within the character set is being represented.

Each function set is associated either with area CL or with area CR, and can be mapped only onto the area with which it is associated. It follows that a given control function from a particular function set cannot at one time have one representation and at another time have some other representation.

2.15 Token set shapes

Areas GL and GR can be said to have the same shape, as they both contain six columns and 16 rows. By contrast areas CL and CR both contain two columns and 16 rows.

A token set can be said to have a shape, which is related to the shape of the area/areas which is/are able to represent that token set.

The above would imply two *token set shapes*: six columns of 16 rows, or two columns of 16 rows. There are in fact three further token set shapes, making five shapes in all. The five shapes are:

- two columns of 16 cells,
- six columns of 16 cells,
- six columns of 16 cells, with the first cell of the first column and the last cell of the last column omitted,
- a set of *n* planes (where *n* is greater than one), each plane containing six columns of 16 cells,
- a set of *n* planes (where *n* is greater than one), each plane containing six columns of 16 cells, with the first cell of the first column and the last cell of the last column omitted in every plane.

The first of the above shapes is used exclusively for function sets; the remaining four shapes are used exclusively for character sets.

The first three shapes each use a single byte to represent a token. The last two shapes each use a sequence of *n* bytes to represent a token. It follows that the number of tokens that can be represented by each of the five token set shapes is, respectively:

- 32 tokens,
- 96 tokens,
- 94 tokens,
- 96^n tokens, and
- 94^n tokens.

The last two of the above token set shapes can clearly accommodate large numbers of tokens. If *n* is, for instance, equal to two, then 96^n is 9216 and 94^n is 8836.

The various token set shapes are illustrated by figure 2.3.

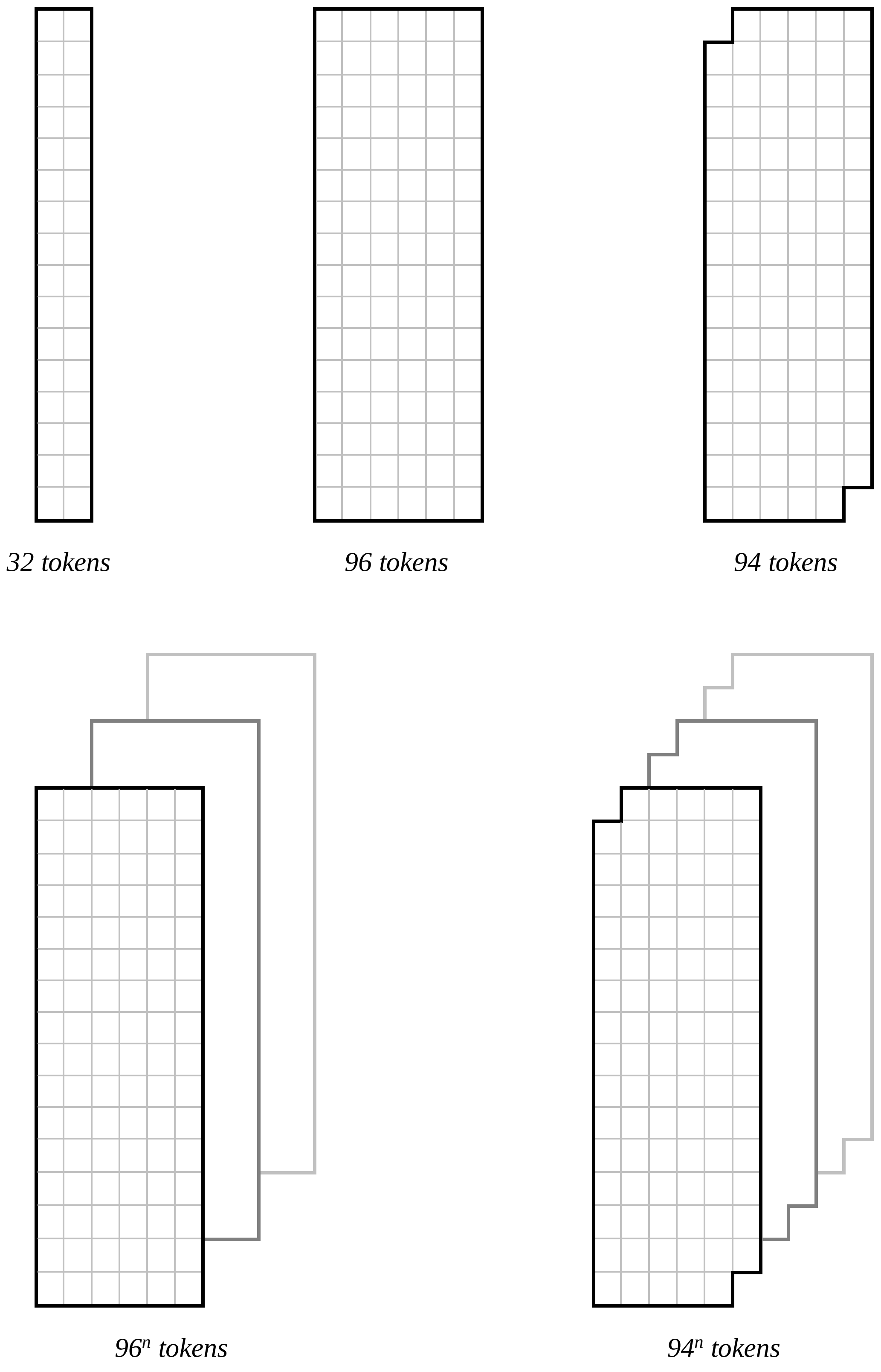


Figure 2.3 – Token set shapes

2.16 Repertoires

Token sets are organised into *repertoires*. Though there are five basic token set shapes, there are six repertoires. This is caused by the division of function sets (ie token sets of the first shape listed above) into those associated with area CL and those associated with area CR.

The six token set repertoires are:

- repertoire C0, containing sets of 32 functions associated with area CL,
- repertoire C1, containing sets of 32 functions associated with area CR,
- repertoire 96, containing sets of 96 characters associated with areas GL and GR,
- repertoire 94, containing sets of 94 characters associated with areas GL and GR,
- repertoire 96^n, containing sets of 96^n characters associated with areas GL and GR, and
- repertoire 94^n, containing sets of 94^n characters associated with areas GL and GR.

The first four repertoires contain *single-byte* token sets, requiring a single byte to represent each token. The last two repertoires contain *multi-byte* token sets, requiring *n* bytes to represent each token.

2.17 Partially populated token sets

A token set groups together a number of related tokens. Such related tokens may not fall naturally into groups of, say, 8836 (the number of available positions in a character set from repertoire 94^2). For this reason, some token sets are partially populated. In the case of a *partially populated token set*, some otherwise legal byte sequences do not represent any tokens.

A partially populated token set is considered to be of the shape it would have if fully populated.

Both character sets and function sets may be partially populated.

2.18 Working sets

The function sets in repertoires C0 and C1 are mapped directly, by use of the appropriate mapping functions, onto areas CL and CR (respectively).

Character sets, on the other hand, are not mapped directly onto areas GL and GR. Their mapping is indirect and makes use of four *working sets*. The four working sets (named G0, G1, G2 and G3) are not character sets in their own right.

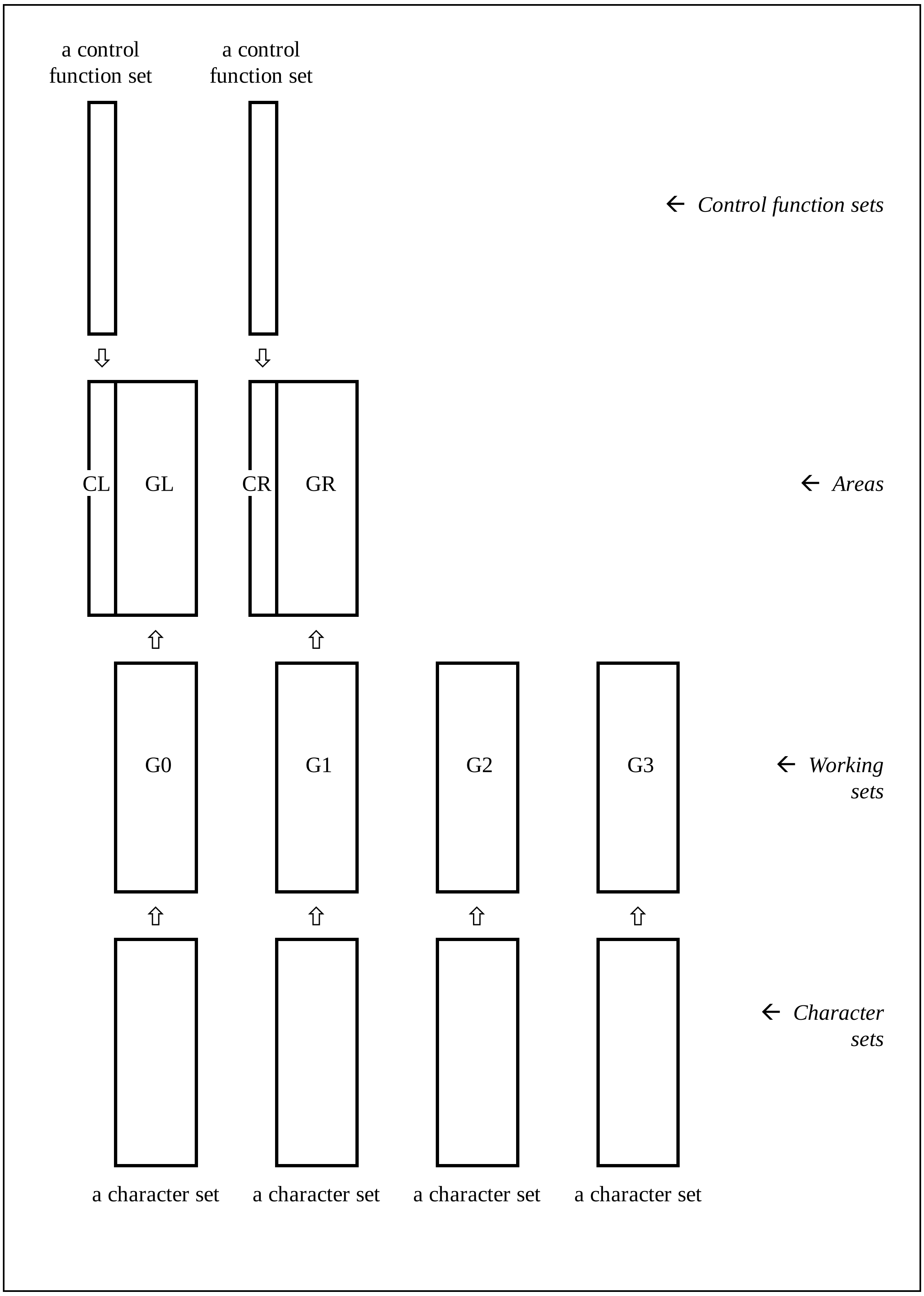
2.19 Current context

At any given time, one character set is mapped onto working set G0, one character set is mapped onto working set G1, one character set is mapped onto working set G2, and one character set is mapped onto working set G3.

At any given time, one of the four working sets is mapped onto area GL, and one of the four working sets is mapped onto area GR.

At any given time, one function set is mapped onto area CL, and one function set is mapped onto area CR.

All the above mappings, taken together, constitute the current context. A graphical representation of the current context is provided in figure 2.4.



Arrows of the form ⇧ and ⇩ represent the various mappings which together make up the current context.

In this figure the boxes representing the character sets are symbolic and do not indicate real token set shapes.

Figure 2.4 – Current context

2.20 Mapping functions

The process of mapping a character set onto a working set is called *designation*.

The process of mapping a working set onto an area is called *invocation*.

The process of mapping a function set onto an area is called *selection*. It combines in one action both designation and invocation.

The current context defines each designation, invocation, and selection currently in force. Mapping functions are those functions which modify the current context. A mapping function may be:

- a character set *designation function*,
- a working set *invocation function*, or
- a function set *selection function*.

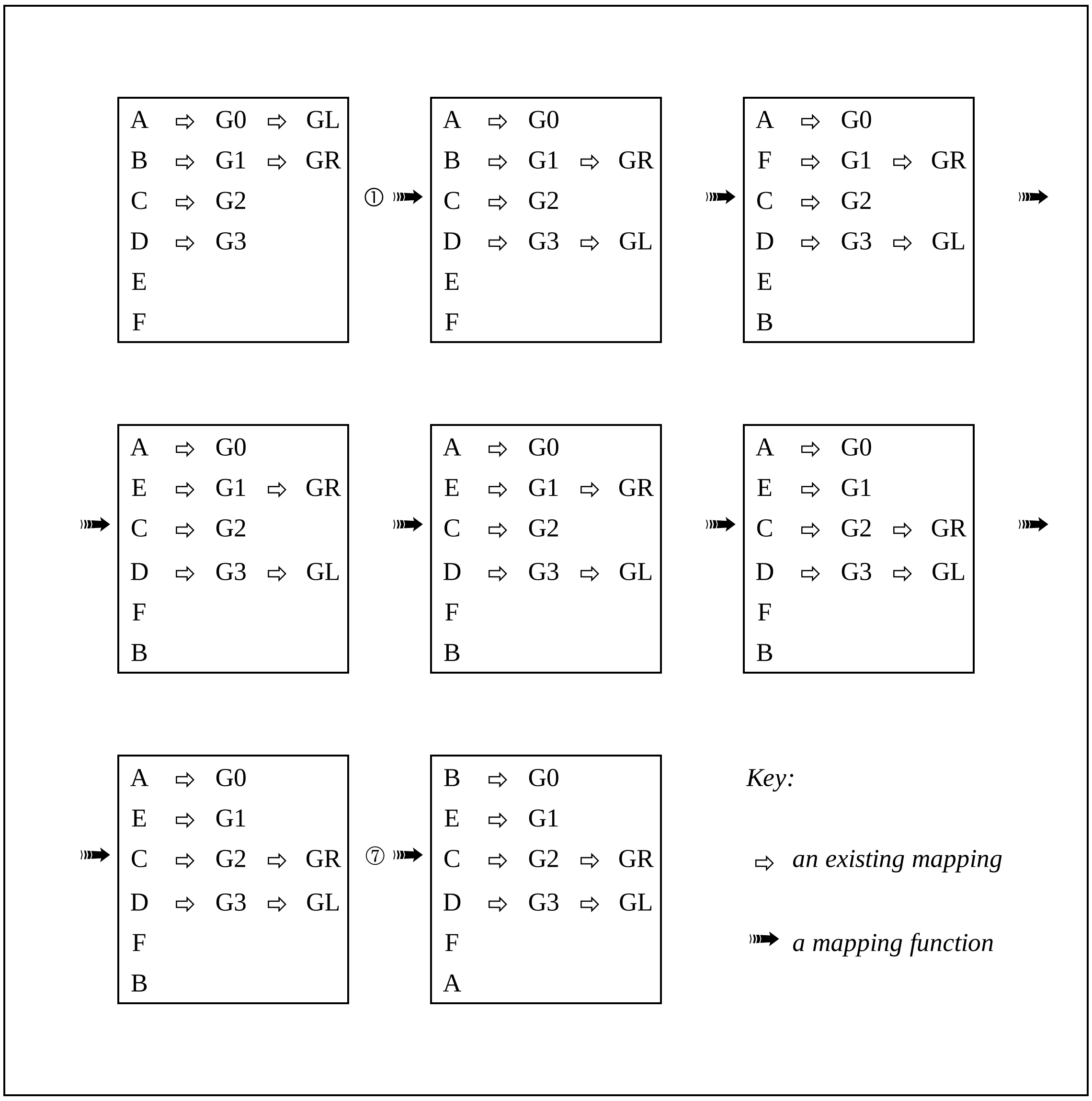
In the case of character sets, designation and invocation are completely independent of one another. This means that an invocation which is in place prior to the execution of a designation function remains in place after the execution of that designation function. Conversely, a designation which is in place prior to the execution of an invocation function remains in place after the execution of that invocation function. This point is illustrated by figure 2.5.

2.21 Restrictions

In general, any character set may be designated to any working set, and any working set may be invoked to either area.

There are two exceptions to the above statement, both imposed by ISO 2022:

- No character set from repertoire 96 or repertoire 96^n may be designated to working set G0.
- Working set G0 may not be invoked to area GR.



Each box represents a current context (note that function sets have been omitted in order to save space).

The available character sets are called A ... F.

The numbered arrows represent the following mapping functions:

① ➠ *Invoke working set G3 to area GL.*
② ➠ *Designate character set F to working set G1.*
③ ➠ *Designate character set E to working set G1.*
④ ➠ *Invoke working set G1 to area GR.*
⑤ ➠ *Invoke working set G2 to area GR.*
⑥ ➠ *Designate character set C to working set G2.*
⑦ ➠ *Designate character set B to working set G0.*

Note that mapping functions ④ and ⑥ left the current context unchanged.

Figure 2.5 – Effect of successive designations and invocations on the current context

2.22 Designation functions

A given designation function maps a particular character set onto a particular working set. A distinct designation function is defined for each working set to which a character set may be designated.

The designation functions supported by this Standard are given in appendix D.

Some of the designation functions supported by this Standard have been taken from the International Register, while others have been defined by Reuters. Full details are provided in appendix D.

For historical reasons, some of the designation functions listed in appendix D have *alternate functions* whose effect is identical. If a given application is to support a particular character set, it must correctly interpret *all* designation functions listed in appendix D for that character set.

Designation functions are represented by escape sequences – some by a single escape sequence, and others by two escape sequences immediately following one another. Figure 2.6 shows two examples of designation functions.

```
ESC   02/04   02/08   04/07
```

```
ESC   02/06   04/00   ESC   02/04   04/02
```

Figure 2.6 – Examples of designation functions

2.23 Invocation (shift) functions

A given invocation function maps a particular working set onto a particular area. For historical reasons, invocation functions are more generally known as shift functions.

Shift functions divide into two categories:

- *locking-shift functions*, and
- *single-shift functions*.

The mapping caused by a locking-shift function remains in place until overridden by some other locking-shift function. Hence the mapping is called a *locking-shift*.

At any one time, two locking-shifts are in place. One defines the working set mapped onto area GL, the other the working set mapped onto area GR.

Though there are four working sets and two areas, there are only seven locking-shift functions, rather than eight. This is due to the absence of a locking-shift function mapping working set G0 to area GR. ISO 2022 does not permit this mapping.

The mapping caused by a single-shift function remains in place only for the processing of a single byte sequence (representing a single character). The *single-shift* is then automatically cancelled, and the current context returns to the state it had prior to the single-shift function.

There are only two single-shift functions:

- a function mapping a single character from working set G2 to area GL, and
- a function mapping a single character from working set G3 to area GL.

The top bit of every byte in the byte sequence following a single-shift function is always set to ZERO.

Locking-shift functions and single-shift functions are wholly independent of one another. This means that a locking-shift which is in place prior to a single-shift function remains in place after the end of the single-shift function. Conversely, the current state of the locking-shifts has no effect on the behaviour of a single-shift function.

The various shift functions are listed in figures 2.7 and 2.8, together with the byte sequences used to represent them.

2.24 Selection functions

A given selection function maps a particular function set onto the area with which that function set is associated.

The selection functions supported by this Standard are given in appendices B and C.

Some of the selection functions supported by this Standard have been taken from the International Register, while others have been defined by Reuters. Full details are provided in appendices B and C.

For historical reasons, some of the selection functions listed in appendices B and C have *alternate functions* whose effect is identical. If a given application is to support a particular function set, it must correctly interpret *all* selection functions listed in appendix B or C for that function set.

Selection functions are represented by escape sequences – some by a single escape sequence, and others by two escape sequences immediately following one another.

Locking-shift function	*Acronym*	*Working set*		*Representation*	
		invoked	*to area*	*decimal*	*hexadecimal*
LOCKING-SHIFT ZERO	LS0	G0	GL	00/15	0F
LOCKING-SHIFT ONE	LS1	G1	GL	00/14	0E
LOCKING-SHIFT ONE RIGHT	LS1R	G1	GR	ESC 07/14	ESC 7E
LOCKING-SHIFT TWO	LS2	G2	GL	ESC 06/14	ESC 6E
LOCKING-SHIFT TWO RIGHT	LS2R	G2	GR	ESC 07/13	ESC 7D
LOCKING-SHIFT THREE	LS3	G3	GL	ESC 06/15	ESC 6F
LOCKING-SHIFT THREE RIGHT	LS3R	G3	GR	ESC 07/12	ESC 7C

Figure 2.7 – Locking-shift functions

Single-shift function	*Acronym*	*Single character*		*Representation*	
		invoked from working set	*to area*	*decimal*	*hexadecimal*
SINGLE-SHIFT TWO	SS2	G2	GL	08/14	8E
SINGLE-SHIFT THREE	SS3	G3	GL	08/15	8F

Figure 2.8 – Single-shift functions

2.25 The missing corners

In figure 2.1 four cells are shown shaded – grey in the case of area GL, and black in the case of area GR. They are the top-left and the bottom-right cell of each area. These four *special cells* are treated as follows:

- When a character set from repertoire 96 or repertoire 96^n is mapped onto either area, the behaviour of the two special cells in that area is no different from the behaviour of the other 94 cells in the area. Byte values corresponding to all 96 cells are used to represent all or part of characters in the relevant character set.

- When a character set from repertoire 94 or repertoire 94^n is mapped onto area GR, the two special cells in that area are treated differently from the other 94 cells in that area. Byte values corresponding to these two cells cannot be used for any purpose while such a mapping is in force.

- When a character set from repertoire 94 or repertoire 94^n is mapped onto area GL, the two special cells in that area are also treated differently from the other 94 cells in that area. In this case, the two special cells in area GL each represent a special character which is not part of any character set. These special characters are:

  - SPACE, abbreviated to SP, and represented by byte value 20 (02/00),

  - DELETE, abbreviated to DEL, and represented by byte value 7F (07/15).

SPACE is a character whose visual representation consists of the absence of a graphic symbol. It causes the active position to be advanced by one character position.

DELETE is a character which is used to indicate an erroneous or unwanted character. The visual representation of DELETE is a grey block occupying one character position (eg ▒). If implementation reasons preclude this visual representation, then the visual representation of the character QUESTION MARK may be used instead.

Each of these two characters is represented by a single byte, irrespective of the number of bytes used to represent each character in the particular character set currently mapped onto area GL.

For instance, if a character set from repertoire 94^2 is mapped onto area GL, then 20 20 (02/00 02/00) represents two SPACE characters and 7F 7F (07/15 07/15) represents two DELETE characters.

The characters SPACE and DELETE cannot be represented while a character set from repertoire 96 or repertoire 96^n is mapped onto area GL.

2.26 Treatment of Nulls

NULL, abbreviated to NUL, is a control function represented by byte value 00 (00/00).

Trailing NULL bytes occurring within a field should be treated as padding and be discarded.

2.27 Permanently assigned values

Most of the 256 possible byte values have no fixed meaning. Mapping functions determine which tokens these values, taken either alone or in combination with other values, represent.

A few of the 256 byte values, however, permanently represent specific tokens. They do so irrespective of the current context and are not affected by mapping functions. These permanently assigned byte values are shown in figure 2.9. They all lie in area CL or area CR.

Byte value		*Area*	*Token name*	*Acronym*
decimal	*hexadecimal*			
00/00	00	CL	NULL	NUL
00/14	0E	CL	LOCKING-SHIFT ONE	LS1
00/15	0F	CL	LOCKING-SHIFT ZERO	LS0
01/11	1B	CL	ESCAPE	ESC
08/14	8E	CR	SINGLE-SHIFT TWO	SS2
08/15	8F	CR	SINGLE-SHIFT THREE	SS3

Figure 2.9 – Permanently assigned byte values

From the above, it follows that:

- every control function set in repertoire C0 contains NUL, LS1, LS0 and ESC, and
- every control function set in repertoire C1 contains SS2 and SS3,

at the positions shown in figure 2.9.

2.28 Information interchange

At the beginning of each information interchange the current context is set to the initial context, which is defined in appendix E. During the information interchange, mapping functions cause cumulative changes to the current context.

Each information interchange corresponds to a single *field* of a Reuter record, and the beginning of each information interchange corresponds to the start of a field. This means that the current context is reset to the initial context at the start of every field.

2.29 Field length

The *field length* is defined as the number of bytes in the byte stream which together represent the text in that field. This is not, in general, the same as the number of characters nor the number of display positions occupied by that text.

2.30 Conformance

Conformance to this Standard divides into three parts:

A conformance to the Reuter Multilingual Text Encoding Scheme, including the sections on error processing,

B correct handling of character sets supported by this Standard, and

C correct handling of function sets supported by this Standard.

Part A

This part is mandatory and includes the correct interpretation of all shift functions.

Part B

This part requires an application to handle fully and correctly those character sets which are relevant to the application's purpose. This includes the correct interpretation of *all* designation functions for those character sets.

The application is not required to interpret designation functions for character sets which are not relevant to it's purpose.

Part C

This part requires an application to handle correctly those function sets, and those individual control functions within those function sets, which are relevant to the application's purpose. This includes the correct interpretation of alternate selection functions for the same function set.

The application is not required to interpret:

- individual control functions which are not relevant to it's purpose, and
- function sets which are not relevant to it's purpose.

2.31 Error processing

Errors of many kinds may be detected in the incoming byte stream. The possible errors are grouped below by their severity, which determines how they are to be handled. The categories used are:

- major errors,
- minor errors, and
- not errors.

The last category is included in order to highlight the correct treatment of certain events which could be thought of as errors but are not.

2.32 Major errors

The major errors are listed in groups, owing to their large number.

The italicised portions provide additional detail.

One of the aims of this section has been the avoidance of any ambiguity. The detailed enumeration of all possible error conditions has led to a somewhat repetitive style. The manner of presentation used should not be taken as an indication of a proposed software design.

Errors detected during the processing of an escape sequence (or of two escape sequences immediately following one another):

➢ An escape sequence (or two escape sequences immediately following one another) not supported by the application is received.

The byte-by-byte parsing of an escape sequence terminates in one of four ways. The only correct termination is:

- *the accumulated bytes are found to match a known escape sequence.*

The three erroneous terminations are:

- *an end of field is encountered,*
- *a byte value is encountered which may not legally appear within an escape sequence,*
- *a byte value is encountered which (though legal), when considered together with the other accumulated bytes, prevents any match with a known escape sequence.*

Errors involving a single-shift function:

- ➢ Immediately following a single-shift function, an end of field is encountered.
- ➢ While a character set from repertoire 96 or repertoire 96^n is designated to working set G2, the single-shift function SS2 is received, and is not immediately followed by a byte sequence consisting of one byte (in the case of repertoire 96) or of *n* bytes (in the case of repertoire 96^n) with value(s) in the range 20 to 7F.
- ➢ While a character set from repertoire 94 or repertoire 94^n is designated to working set G2, the single-shift function SS2 is received, and is not immediately followed by a byte sequence consisting of one byte (in the case of repertoire 94) or of *n* bytes (in the case of repertoire 94^n) with value(s) in the range 21 to 7E.
- ➢ While a character set from repertoire 96 or repertoire 96^n is designated to working set G3, the single-shift function SS3 is received, and is not immediately followed by a byte sequence consisting of one byte (in the case of repertoire 96) or of *n* bytes (in the case of repertoire 96^n) with value(s) in the range 20 to 7F.
- ➢ While a character set from repertoire 94 or repertoire 94^n is designated to working set G3, the single-shift function SS3 is received, and is not immediately followed by a byte sequence consisting of one byte (in the case of repertoire 94) or of *n* bytes (in the case of repertoire 94^n) with value(s) in the range 21 to 7E.

Errors involving a character from repertoire 96^n or repertoire 94^n but not involving a single-shift function:

- ➢ Part of the way through a byte sequence representing a character from a character set in repertoire 96^n or repertoire 94^n, an end of field is encountered.
- ➢ While a character set from repertoire 96^n is mapped onto area GL, the first byte of a byte sequence representing a character from that character set is received, but is not immediately followed by *n-1* bytes in the range 20 to 7F.
- ➢ While a character set from repertoire 94^n is mapped onto area GL, the first byte of a byte sequence representing a character from that character set is received, but is not immediately followed by *n-1* bytes in the range 21 to 7E.
- ➢ While a character set from repertoire 96^n is mapped onto area GR, the first byte of a byte sequence representing a character from that character set is received, but is not immediately followed by *n-1* bytes in the range A0 to FF.

➢ While a character set from repertoire 94^n is mapped onto area GR, the first byte of a byte sequence representing a character from that character set is received, but is not immediately followed by *n-1* bytes in the range A1 to FE.

Other errors:

➢ While a character set from repertoire 94 or repertoire 94^n is mapped onto area GR, either of the two byte values A0 or FF is received.

➢ A byte sequence representing an individual control function not supported by the application is received.

➢ A byte sequence representing an unpopulated position in a function set is received.

All major errors are handled by discarding all unprocessed bytes intended for the current field, and proceeding to the next field. This, at first sight drastic step, is necessary in order to reach a known state. Errors in this category cause a loss of synchronisation between the producing and the consuming applications. The resetting of the current context to the initial context which occurs at the start of the next field restores the lost synchronisation.

2.33 Minor errors

The following is a minor error:

➢ A byte sequence representing an unpopulated position in a character set is received.

This minor error is handled by:

- discarding *only the byte sequence representing that single character*,
- treating the discarded byte sequence as representing a single character whose visual representation is a grey block occupying one character position (eg ▒), and
- continuing to process any subsequent bytes representing text in the current field.

If implementation reasons preclude the above visual representation, then the visual representation of the character QUESTION MARK may be used instead.

Note that the same grey block is also used as the visual representation of the character DELETE.

2.34 Not errors

The following are not errors:

- A function designating a character set to a working set to which it is currently designated is received.
- A function invoking a working set to an area to which it is currently invoked is received.
- A function selecting a function set which is currently selected is received.

The above events are handled by either processing or discarding the received function – the effect of the two approaches is the same.

A Glossary

For the purpose of this Standard, the following definitions apply.

alternate function ~ For historical reasons, some of the designation and selection functions defined by this Standard have alternate functions whose effect is identical. If a given application is to support a particular character set, it must correctly interpret all designation functions listed in this Standard for that character set. If a given application is to support a particular function set, it must correctly interpret all selection functions listed in this Standard for that function set.

area ~ The matrix is divided into four areas named CL, CR, GL and GR. Areas CL and CR are used for the representation of control functions. Areas GL and GR are used for the representation of graphic characters.

beginning of an information interchange ~ Each information interchange corresponds to a single field of a Reuter record, and the beginning of each information interchange corresponds to the start of a field.

byte ~ A byte is an ordered set of eight bits, treated as a unit.

byte sequence ~ A byte sequence is an ordered set of one or more bytes used to represent a token.

byte stream ~ The succession of tokens passed from producing application to consuming application is represented by a succession of byte sequences. The ordered set of bytes making up all of these byte sequences is the byte stream.

byte value ~ A byte value is one of the 256 possible values of a byte. Byte values are the building blocks used to construct legal byte sequences, which in turn represent legal tokens.

cell ~ The matrix is divided into 256 cells. Each of these cells is equivalent to one of the 256 possible byte values.

character ~ See *graphic character*.

character set ~ A character set is a set of graphic characters grouped together for some purpose.

control function ~ A control function is any function which is not a mapping function. An example of a control function is CARRIAGE RETURN.

control function set ~ A control function set is a set of control functions grouped together for some purpose.

current context ~ At any given time, one character set is mapped onto working set G0, one character set is mapped onto working set G1, one character set is mapped onto working set G2, and one character set is mapped onto working set G3. At any given time, one of the four working sets is mapped onto area GL, and one of the four working sets is mapped onto area GR. At any given time, one function set is mapped onto area CL, and one function set is mapped onto area CR. All these mappings, taken together, constitute the current context.

designation ~ The process of mapping a character set onto a working set is called designation.

designation function ~ A given designation function maps a particular character set onto a particular working set. A distinct designation function is defined for each working set to which a character set may be designated.

ECMA ~ ECMA, the European Computer Manufacturers Association, has been designated by ISO as the International Registration Authority tasked with maintaining the *International Register of Coded Character Sets to be used with Escape Sequences*.

escape sequence ~ An escape sequence is a byte sequence which starts with ESC, and contains one or more additional bytes drawn from area GL. Some (though not all) mapping functions are represented by these special byte sequences.

field ~ A field is a unit of information within a Reuter record. This Standard defines the encoding of text within textual fields.

field length ~ The field length is defined as the number of bytes in the byte stream which together represent the text in that field. This is not, in general, the same as the number of characters nor the number of display positions occupied by that text.

function ~ A function is either a control function or a mapping function.

function set ~ See *control function set.*

graphic character ~ A graphic character is a member of a set of elements used for the representation of data. It has one or more visual representations which may be hand-written, printed or displayed.

information interchange ~ Each information interchange corresponds to a single field of a Reuter record.

initial context ~ At the beginning of each information interchange the current context is set to the initial context, which is defined in appendix E.

International Register ~ See *International Register of Coded Character Sets to be used with Escape Sequences.*

International Register of Coded Character Sets to be used with Escape Sequences ~ ISO 2022 defines the rules which govern the handling of character sets, but does not define the character sets themselves. Instead, it refers to the *International Register of Coded Character Sets to be used with Escape Sequences*, together with an *International Registration Authority* tasked with maintaining this Register. *ECMA*, the European Computer Manufacturers Association, has been designated as the International Registration Authority.

International Registration Authority ~ See *International Register of Coded Character Sets to be used with Escape Sequences.*

invocation ~ The process of mapping a working set onto an area is called *invocation*.

invocation function ~ A given invocation function maps a particular working set onto a particular area. For historical reasons, invocation functions are more generally known as shift functions.

locking-shift ~ The mapping caused by a locking-shift function remains in place until overridden by some other locking-shift function. Hence the mapping is called a locking-shift.

locking-shift function ~ An invocation (shift) function whose effect lasts until overridden by some other locking-shift function.

mapping ~ Token sets may be mapped onto various of the four areas. When a token set is mapped onto an area, byte values within that area are used to construct byte sequences representing tokens in that token set.

mapping function ~ The current context defines each designation, invocation, and selection currently in force. Mapping functions are those functions which modify the current context. A mapping function may be a character set designation function, a working set invocation function, or a function set selection function.

matrix ~ The 256 distinct byte values are the building blocks used to construct legal byte sequences, which in turn represent legal tokens. The 256 byte values are shown graphically as a 16 x 16 matrix. Each of the 256 cells within the matrix is equivalent to one of the 256 possible byte values.

meaning ~ In general, the cells of the matrix do not have permanently assigned meanings. Their meanings depend upon the current context, and change through the use of mapping functions.

multi-byte ~ Four of the six repertoires contain single-byte token sets, requiring a single byte to represent each token. Two of the repertoires contain multi-byte token sets, requiring *n* bytes to represent each token.

n ~ *n* represents an integer greater than one.

partially populated token set ~ In the case of a partially populated token set, some otherwise legal byte sequences do not represent any tokens.

Register ~ See *International Register of Coded Character Sets to be used with Escape Sequences*.

repertoire ~ Token sets are organised into repertoires. The six repertoires are: repertoire C0, repertoire C1, repertoire 96, repertoire 94, repertoire 96^n, and repertoire 94^n.

representation ~ Each token in a set is represented, within an appropriate context, by some byte sequence.

Reuter Multilingual Text Encoding Scheme ~ The Reuter Multilingual Text Encoding Scheme is the Reuter Multilingual Text Encoding Standard minus the appendices describing the particular character sets and function sets supported by the Standard.

Reuter Multilingual Text Encoding Standard ~ The Reuter Multilingual Text Encoding Standard is this document.

selection ~ The process of mapping a function set onto an area is called selection. It combines in one action both designation and invocation.

selection function ~ A given selection function maps a particular function set onto the area with which that function set is associated.

shift function ~ See *invocation function*.

single-byte ~ Four of the six repertoires contain single-byte token sets, requiring a single byte to represent each token. Two of the repertoires contain multi-byte token sets, requiring *n* bytes to represent each token.

single-shift ~ The mapping caused by a single-shift function remains in place only for the processing of a single byte sequence (representing a single character). The single-shift is then automatically cancelled, and the current context returns to the state it had prior to the single-shift function.

single-shift function ~ An invocation (shift) function whose effect lasts only for the processing of a single byte sequence (representing a single character).

special cell ~ The four special cells correspond to byte values 20 (02/00), 7F (07/15), A0 (10/00) and FF (15/15).

token ~ The Reuter Multilingual Text Encoding Standard defines a set of rules governing the interchange of multilingual textual information. This information is made up of tokens. A given token may be a graphic character, a control function, or a mapping function.

token set ~ Both graphic characters and control functions are organised into sets of related tokens. Such a token set is either a character set or a function set.

token set shape ~ There are five token set shapes: (i) two columns of 16 cells, (ii) six columns of 16 cells, (iii) six columns of 16 cells, with the first cell of the first column and the last cell of the last column omitted, (iv) a set of *n* planes (where *n* is greater than one), each plane containing six columns of 16 cells, and (v) a set of *n* planes (where *n* is greater than one), each plane containing six columns of 16 cells, with the first cell of the first column and the last cell of the last column omitted in every plane.

top bit ~ The top bit is the most significant bit of a byte.

working set ~ Character sets are not mapped directly onto areas GL and GR. Their mapping is indirect and makes use of four working sets. The four working sets (named G0, G1, G2 and G3) are not character sets in their own right.

B Summary of supported control function sets for area CL

This appendix briefly describes the control function sets for area CL currently supported by this Standard. For additional detail see appendices F and G.

In the following tables the information in fields:

- Name of set,
- Origin,
- Sponsor,
- Registration number,
- Registration date, and
- Pages in Register (ie page number reference to Register)

is, in the case of internationally registered function sets, taken from the *International Register of Coded Character Sets to be used with Escape Sequences*, for which ISO has designated ECMA, the European Computer Manufacturers Association, as the International Registration Authority.

Those selection functions which have been taken from the International Register have been marked with the letters *IS*, standing for International Standard. Other selection functions been allocated by Reuters and have been indicated with the letters *RS*, standing for Reuter Standard.

B.1 Reuter basic control function set 1

Name of set	The set of control characters of ISO 646	
Origin	ISO 646	
Sponsor	Secretariat ISO/TC97/SC2	
Registration number	1	
Registration date	1 December 1975	
Pages in Register	4.2 – 4.8	
Bytes per character	1	
Shape of set	32	
Number of characters	32	
Empty positions	0	
Description	This set of control functions, listed in ISO 646 and defined in ISO 6429, is the set of functions generally associated with the ASCII character set	
CL selection	ESC 02/01 04/00	*IS*

C Summary of supported control function sets for area CR

This appendix briefly describes the control function sets for area CR currently supported by this Standard. For additional detail see appendices F and G.

The introduction to appendix B explains the notation used below.

C.1 Reuter basic control function set 2

Name of set	Reuter basic control function set 2	
Origin	This Standard	
Sponsor	Reuters	
Registration number	–	
Registration date	–	
Pages in Register	–	
Bytes per character	1	
Shape of set	32	
Number of characters	24	
Empty positions	8	
Description	A subset of the set of control functions defined in ISO 6429 as the usual C1 set	
CR selection	ESC 02/02 03/00	*RS*

D Summary of supported character sets

This appendix briefly describes the character sets currently supported by this Standard. The first two of the character sets listed below are reproduced in this document (see appendices F and G).

Readers who require copies of the additional character sets may obtain the document *Reuter Multilingual Text Encoding Standard, Supplement 1 ~ Additional character sets* from the Reuter Publications Department. That document contains all character sets which are supported by this Standard but are not reproduced here. Alternatively, readers may obtain individual character sets by contacting ECMA, 114 Rue du Rhône, CH-1204 Geneva, Switzerland (☎ + 41 22 735 36 34) and quoting the appropriate character set Registration number.

In the following tables the information in fields:

- Name of set,
- Origin,
- Sponsor,
- Registration number,
- Registration date, and
- Pages in Register (ie page number reference to Register)

is, in the case of internationally registered function sets, taken from the *International Register of Coded Character Sets to be used with Escape Sequences*, for which ISO has designated ECMA, the European Computer Manufacturers Association, as the International Registration Authority.

Most of the designation functions listed in this appendix have been taken from the International Register. These have been indicated with the letters *IS*, standing for International Standard. Other designation functions been allocated by Reuters and have been indicated with the letters *RS*, standing for Reuter Standard. Those instances where neither ECMA nor Reuters have allocated a designation function are marked with the character "–".

D.1 Reuter basic character set 1

Name of set	ASCII graphic character set	
Origin	American National Standard X3.4 – 1968	
Sponsor	American National Standards Institute	
Registration number	6	
Registration date	1 December 1975	
Pages in Register	3.14 – 3.19	
Bytes per character	1	
Shape of set	94	
Number of characters	94	
Empty positions	0	
Description	A character set based on the ISO 646 set, with some small differences such as a Dollar sign placed in position 24	
G0 designation	ESC 02/08 04/02	*IS*
G1 designation	ESC 02/09 04/02	*IS*
G2 designation	–	–
G3 designation	–	–

D.2 Reuter basic character set 2

Name of set	Reuter basic character set 2	
Origin	This Standard	
Sponsor	Reuters	
Registration number	–	
Registration date	–	
Pages in Register	–	
Bytes per character	1	
Shape of set	94	
Number of characters	94	
Empty positions	0	
Description	A set of characters chosen by Reuters to complement Reuter basic character set 1, and including some characters specific to Reuters	
G0 designation	–	–
G1 designation	ESC 02/09 03/01	*RS*
G2 designation	–	–
G3 designation	–	–

D.3 Japanese Katakana character set

Name of set	The Japanese Katakana graphic set of characters	
Origin	Japanese Standard JIS C 6220 – 1969	
Sponsor	Japanese Industrial Standards Committee	
Registration number	13	
Registration date	1 December 1975	
Pages in Register	3.56 – 3.60	
Bytes per character	1	
Shape of set	94	
Number of characters	63	
Empty positions	31	
Description	This Japanese Katakana graphic set of characters is identical to the Katakana character set described in the Japanese Standard JIS X 0201 – 1976	
G0 designation	ESC 02/08 04/09	*IS*
G1 designation	ESC 02/09 04/09	*IS*
G2 designation	ESC 02/10 03/02	*RS*
G3 designation	–	–

D.4 Japanese Latin character set

Name of set	The Japanese Roman graphic set of characters	
Origin	Japanese Standard JIS C 6220 – 1969	
Sponsor	Japanese Industrial Standards Committee	
Registration number	14	
Registration date	1 December 1975	
Pages in Register	3.61 – 3.67	
Bytes per character	1	
Shape of set	94	
Number of characters	94	
Empty positions	0	
Description	A character set similar to ASCII (with some small differences such as a Yen sign placed in position 5C) which is identical to the Latin character set described in the Japanese Standard JIS X 0201 – 1976	
G0 designation	ESC 02/08 04/10	*IS*
G1 designation	ESC 02/09 04/10	*IS*
G2 designation	–	–
G3 designation	ESC 02/11 03/03	*RS*

D.5 Japanese Kanji character set

Name of set	Japanese Graphic Character Set for Information Interchange	
Origin	Japanese Standard JIS X 0208 – 1990	
Sponsor	Japanese National Committee on ISO/IEC JTC1/SC2	
Registration number	168	
Registration date	13 July 1992	
Pages in Register	3.587 – 3.600	
Bytes per character	2	
Shape of set	94^2	
Number of characters	6879	
Empty positions	1957	
Description	The character set consists of the following groups of characters: • General graphic characters (147 characters) • Arabic digits (10 characters) • Latin alphabet (52 characters) • Hiragana characters (83 characters) • Katakana characters (86 characters) • Greek alphabet (48 characters) • Cyrillic alphabet (66 characters) • Kanji characters (6355 characters) • Line drawing characters (32 characters)	
G0 designation	ESC 02/06 04/00 ESC 02/04 04/02	*IS*
G1 designation	ESC 02/06 04/00 ESC 02/04 02/09 04/02	*IS*
G2 designation	ESC 02/06 04/00 ESC 02/04 02/10 04/02	*IS*
G3 designation	ESC 02/06 04/00 ESC 02/04 02/11 04/02	*IS*
or	ESC 02/04 02/11 03/04	*RS*

D.6 Chinese character set 1

Name of set	Chinese Standard Interchange Code – Set 1	
Origin	Chinese National Standard CNS 11643 – 1986	
Sponsor	European Computer Manufacturers Association (ECMA)	
Registration number	171	
Registration date	21 January 1993 (corrected 7 July 1994)	
Pages in Register	3.735 – 3.749	
Bytes per character	2	
Shape of set	94^2	
Number of characters	6085	
Empty positions	2751	
Description	The character set consists of the following groups of characters: • General graphic symbols (234 characters) • Numeric symbols (62 characters) • Alphabetic characters (100 characters) • Mandarin phonetic symbols (42 characters) • Chinese character radicals (213 characters) • Control code symbols (33 characters) • Frequently-used Chinese characters (5401 characters)	
G0 designation	ESC 02/04 02/08 04/07	*IS*
G1 designation	ESC 02/04 02/09 04/07	*IS*
G2 designation	ESC 02/04 02/10 04/07	*IS*
or	ESC 02/04 02/10 03/05	*RS*
G3 designation	ESC 02/04 02/11 04/07	*IS*

D.7 Chinese character set 2

Name of set	Chinese Standard Interchange Code – Set 2	
Origin	Chinese National Standard CNS 11643 – 1986	
Sponsor	European Computer Manufacturers Association (ECMA)	
Registration number	172	
Registration date	21 January 1993 (corrected 7 July 1994)	
Pages in Register	3.750 – 3.763	
Bytes per character	2	
Shape of set	94^2	
Number of characters	7650	
Empty positions	1186	
Description	Less-frequently-used Chinese characters	
G0 designation	ESC 02/04 02/08 04/08	*IS*
G1 designation	ESC 02/04 02/09 04/08	*IS*
G2 designation	ESC 02/04 02/10 04/08	*IS*
G3 designation	ESC 02/04 02/11 04/08	*IS*
or	ESC 02/04 02/11 03/06	*RS*

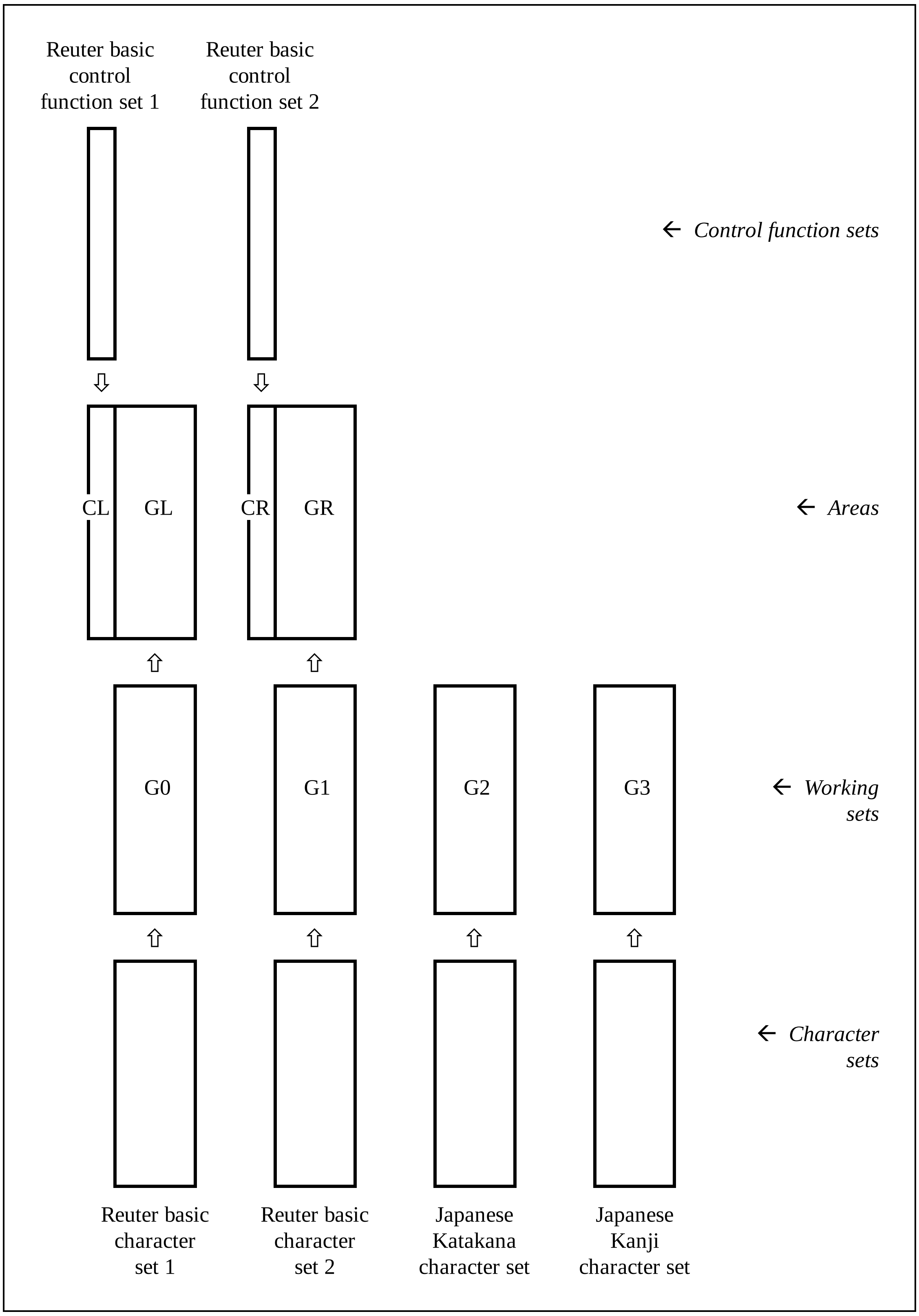
E Initial context

This appendix defines the initial context which applies at the beginning of every information interchange. The information tabulated below is presented graphically in section E.2.

E.1 Tabular presentation

Control function set selected to area CL	Reuter basic control function set 1
Control function set selected to area CR	Reuter basic control function set 2
Character set designated to working set G0	Reuter basic character set 1
Character set designated to working set G1	Reuter basic character set 2
Character set designated to working set G2	Japanese Katakana character set
Character set designated to working set G3	Japanese Kanji character set
Working set invoked to area GL	Working set G0
Working set invoked to area GR	Working set G1

E.2 Graphical presentation


Reuter basic control function set 1
Reuter basic control function set 2
← Control function sets
CL
GL
CR
GR
← Areas
G0
G1
G2
G3
← Working sets
← Character sets
Reuter basic character set 1
Reuter basic character set 2
Japanese Katakana character set
Japanese Kanji character set

Arrows of the form ⇧ and ⇩ represent the various mappings which together make up the initial context.

In this figure the boxes representing the character sets are symbolic and do not indicate real token set shapes.

F Layout of Reuter basic token sets

In the following tables, the outer column and row numbers are decimal, while the inner (italicised) numbers are hexadecimal.

In the case of character sets, the column numbers shown exclude the top bit, which is dynamically determined by the mapping of the character set onto area GL or area GR.

In the case of function sets, however, the column numbers shown include the top bit, as these sets are each associated with either area CL or with area CR, and cannot be mapped onto the other area.

F.1 Reuter basic control function set 1

The introduction to this appendix explains the notation used below.

		00	01
		0	*1*
00	*0*	NUL	DLE
01	*1*	SOH	DC1
02	*2*	STX	DC2
03	*3*	ETX	DC3
04	*4*	EOT	DC4
05	*5*	ENQ	NAK
06	*6*	ACK	SYN
07	*7*	BEL	ETB
08	*8*	BS	CAN
09	*9*	HT	EM
10	*A*	LF	SUB
11	*B*	VT	ESC
12	*C*	FF	IS4 (FS)
13	*D*	CR	IS3 (GS)
14	*E*	LS1	IS2 (RS)
15	*F*	LS0	IS1 (US)

F.2 Reuter basic control function set 2

The introduction to this appendix explains the notation used below.

The shaded cells are not populated.

		08	09
		8	*9*
00	*0*		DCS
01	*1*		PU1
02	*2*		PU2
03	*3*		STS
04	*4*		CCH
05	*5*	NEL	MW
06	*6*	SSA	SPA
07	*7*	ESA	EPA
08	*8*	HTS	
09	*9*	HTJ	
10	*A*	VTS	
11	*B*	PLD	CSI
12	*C*	PLU	ST
13	*D*	RI	OSC
14	*E*	SS2	PM
15	*F*	SS3	APC

F.3 Reuter basic character set 1

The introduction to this appendix explains the notation used below.

		02	03	04	05	06	07
		2	*3*	*4*	*5*	*6*	*7*
00	*0*		0	@	P	`	p
01	*1*	!	1	A	Q	a	q
02	*2*	"	2	B	R	b	r
03	*3*	#	3	C	S	c	s
04	*4*	$	4	D	T	d	t
05	*5*	%	5	E	U	e	u
06	*6*	&	6	F	V	f	v
07	*7*	'	7	G	W	g	w
08	*8*	(	8	H	X	h	x
09	*9*	)	9	I	Y	i	y
10	*A*	*	:	J	Z	j	z
11	*B*	+	;	K	[	k	{
12	*C*	,	<	L	\	l	|
13	*D*	-	=	M	]	m	}
14	*E*	.	>	N	^	n	~
15	*F*	/	?	O	_	o	

F.4 Reuter basic character set 2

The introduction to this appendix explains the notation used below.

		02	03	04	05	06	07
		2	*3*	*4*	*5*	*6*	*7*
00	*0*		°	À	$\mathrm{P_R}$	à	$\mathrm{U_N}$
01	*1*	¡	±	Á	Ñ	á	ñ
02	*2*	¢	^{2}	Â	Ò	â	ò
03	*3*	£	^{3}	Ã	Ó	ã	ó
04	*4*	$\mathrm{R_T}$	$\mathrm{W_S}$	Ä	Ô	ä	ô
05	*5*	¥	µ	Å	Õ	å	õ
06	*6*	$\mathrm{W_I}$	¶	Æ	Ö	æ	ö
07	*7*	§		Ç	Œ	ç	œ
08	*8*	¤	🔔	È	Ø	è	ø
09	*9*	©	^{1}	É	Ù	é	ù
10	*A*	ª	º	Ê	Ú	ê	ú
11	*B*	«	»	Ë	Û	ë	û
12	*C*	⅛	¼	Ì	Ü	ì	ü
13	*D*	⅜	½	Í	Ÿ	í	ÿ
14	*E*	⅝	¾	Î	↑	î	↓
15	*F*	⅞	¿	Ï	β	ï	

G Detailed listing of Reuter basic token sets

In the case of character sets, the column numbers shown in this appendix exclude the top bit, which is dynamically determined by the mapping of the character set onto area GL or area GR.

In the case of function sets, however, the column numbers shown include the top bit, as these sets are each associated with either area CL or with area CR, and cannot be mapped onto the other area.

G.1 Reuter basic control function set 1

The introduction to this appendix explains the notation used below.

Position		*Acronym*	*Name*
decimal	*hex*		
00/00	00	NUL	NULL
00/01	01	SOH	START OF HEADING
00/02	02	STX	START OF TEXT
00/03	03	ETX	END OF TEXT
00/04	04	EOT	END OF TRANSMISSION
00/05	05	ENQ	ENQUIRY
00/06	06	ACK	ACKNOWLEDGE
00/07	07	BEL	BELL
00/08	08	BS	BACKSPACE
00/09	09	HT	CHARACTER TABULATION
00/10	0A	LF	LINE FEED
00/11	0B	VT	LINE TABULATION
00/12	0C	FF	FORM FEED
00/13	0D	CR	CARRIAGE RETURN
00/14	0E	LS1	LOCKING-SHIFT ONE
00/15	0F	LS0	LOCKING-SHIFT ZERO
01/00	10	DLE	DATA LINK ESCAPE
01/01	11	DC1	DEVICE CONTROL ONE
01/02	12	DC2	DEVICE CONTROL TWO
01/03	13	DC3	DEVICE CONTROL THREE
01/04	14	DC4	DEVICE CONTROL FOUR
01/05	15	NAK	NEGATIVE ACKNOWLEDGE
01/06	16	SYN	SYNCHRONOUS IDLE
01/07	17	ETB	END OF TRANSMISSION BLOCK
01/08	18	CAN	CANCEL
01/09	19	EM	END OF MEDIUM
01/10	1A	SUB	SUBSTITUTE
01/11	1B	ESC	ESCAPE
01/12	1C	IS4 (FS)	FILE SEPARATOR
01/13	1D	IS3 (GS)	GROUP SEPARATOR
01/14	1E	IS2 (RS)	RECORD SEPARATOR
01/15	1F	IS1 (US)	UNIT SEPARATOR

G.2 Reuter basic control function set 2

The introduction to this appendix explains the notation used below.

The shaded cells are not populated.

Position		*Acronym*	*Name*
decimal	*hex*		
08/00	80		
08/01	81		
08/02	82		
08/03	83		
08/04	84		
08/05	85	NEL	NEXT LINE
08/06	86	SSA	START OF SELECTED AREA
08/07	87	ESA	END OF SELECTED AREA
08/08	88	HTS	CHARACTER TABULATION SET
08/09	89	HTJ	CHARACTER TABULATION WITH JUSTIFICATION
08/10	8A	VTS	LINE TABULATION SET
08/11	8B	PLD	PARTIAL LINE FORWARD
08/12	8C	PLU	PARTIAL LINE BACKWARD
08/13	8D	RI	REVERSE LINE FEED
08/14	8E	SS2	SINGLE-SHIFT TWO
08/15	8F	SS3	SINGLE-SHIFT THREE
09/00	90	DCS	DEVICE CONTROL STRING
09/09	91	PU1	PRIVATE USE ONE
09/02	92	PU2	PRIVATE USE TWO
09/03	93	STS	SET TRANSMIT STATE
09/04	94	CCH	CANCEL CHARACTER
09/05	95	MW	MESSAGE WAITING
09/06	96	SPA	START OF GUARDED AREA
09/07	97	EPA	END OF GUARDED AREA
09/08	98		
09/09	99		
09/10	9A		
09/11	9B	CSI	CONTROL SEQUENCE INTRODUCER
09/12	9C	ST	STRING TERMINATOR
09/13	9D	OSC	OPERATING SYSTEM COMMAND
09/14	9E	PM	PRIVACY MESSAGE
09/15	9F	APC	APPLICATION PROGRAM COMMAND

G.3 Reuter basic character set 1

The introduction to this appendix explains the notation used below.

Position		*Graphic*	*Name*
decimal	*hex*		
02/01	21	!	EXCLAMATION MARK
02/02	22	"	QUOTATION MARK
02/03	23	#	NUMBER SIGN
02/04	24	$	DOLLAR SIGN
02/05	25	%	PERCENT SIGN
02/06	26	&	AMPERSAND
02/07	27	'	APOSTROPHE
02/08	28	(	LEFT PARENTHESIS
02/09	29	)	RIGHT PARENTHESIS
02/10	2A	*	ASTERISK
02/11	2B	+	PLUS SIGN
02/12	2C	,	COMMA
02/13	2D	-	HYPHEN-MINUS
02/14	2E	.	FULL STOP
02/15	2F	/	SOLIDUS
03/00	30	0	DIGIT ZERO
03/01	31	1	DIGIT ONE
03/02	32	2	DIGIT TWO
03/03	33	3	DIGIT THREE
03/04	34	4	DIGIT FOUR
03/05	35	5	DIGIT FIVE
03/06	36	6	DIGIT SIX
03/07	37	7	DIGIT SEVEN
03/08	38	8	DIGIT EIGHT
03/09	39	9	DIGIT NINE
03/10	3A	:	COLON
03/11	3B	;	SEMICOLON
03/12	3C	<	LESS-THAN SIGN
03/13	3D	=	EQUALS SIGN
03/14	3E	>	GREATER-THAN SIGN
03/15	3F	?	QUESTION MARK

Position		*Graphic*	*Name*
decimal	*hex*		
04/00	40	@	COMMERCIAL AT
04/01	41	A	LATIN CAPITAL LETTER A
04/02	42	B	LATIN CAPITAL LETTER B
04/03	43	C	LATIN CAPITAL LETTER C
04/04	44	D	LATIN CAPITAL LETTER D
04/05	45	E	LATIN CAPITAL LETTER E
04/06	46	F	LATIN CAPITAL LETTER F
04/07	47	G	LATIN CAPITAL LETTER G
04/08	48	H	LATIN CAPITAL LETTER H
04/09	49	I	LATIN CAPITAL LETTER I
04/10	4A	J	LATIN CAPITAL LETTER J
04/11	4B	K	LATIN CAPITAL LETTER K
04/12	4C	L	LATIN CAPITAL LETTER L
04/13	4D	M	LATIN CAPITAL LETTER M
04/14	4E	N	LATIN CAPITAL LETTER N
04/15	4F	O	LATIN CAPITAL LETTER O
05/00	50	P	LATIN CAPITAL LETTER P
05/01	51	Q	LATIN CAPITAL LETTER Q
05/02	52	R	LATIN CAPITAL LETTER R
05/03	53	S	LATIN CAPITAL LETTER S
05/04	54	T	LATIN CAPITAL LETTER T
05/05	55	U	LATIN CAPITAL LETTER U
05/06	56	V	LATIN CAPITAL LETTER V
05/07	57	W	LATIN CAPITAL LETTER W
05/08	58	X	LATIN CAPITAL LETTER X
05/09	59	Y	LATIN CAPITAL LETTER Y
05/10	5A	Z	LATIN CAPITAL LETTER Z
05/11	5B	[	LEFT SQUARE BRACKET
05/12	5C	\	REVERSE SOLIDUS
05/13	5D	]	RIGHT SQUARE BRACKET
05/14	5E	^	CIRCUMFLEX ACCENT
05/15	5F	_	LOW LINE

Position		*Graphic*	*Name*
decimal	*hex*		
06/00	60	`	GRAVE ACCENT
06/01	61	a	LATIN SMALL LETTER A
06/02	62	b	LATIN SMALL LETTER B
06/03	63	c	LATIN SMALL LETTER C
06/04	64	d	LATIN SMALL LETTER D
06/05	65	e	LATIN SMALL LETTER E
06/06	66	f	LATIN SMALL LETTER F
06/07	67	g	LATIN SMALL LETTER G
06/08	68	h	LATIN SMALL LETTER H
06/09	69	i	LATIN SMALL LETTER I
06/10	6A	j	LATIN SMALL LETTER J
06/11	6B	k	LATIN SMALL LETTER K
06/12	6C	l	LATIN SMALL LETTER L
06/13	6D	m	LATIN SMALL LETTER M
06/14	6E	n	LATIN SMALL LETTER N
06/15	6F	o	LATIN SMALL LETTER O
07/00	70	p	LATIN SMALL LETTER P
07/01	71	q	LATIN SMALL LETTER Q
07/02	72	r	LATIN SMALL LETTER R
07/03	73	s	LATIN SMALL LETTER S
07/04	74	t	LATIN SMALL LETTER T
07/05	75	u	LATIN SMALL LETTER U
07/06	76	v	LATIN SMALL LETTER V
07/07	77	w	LATIN SMALL LETTER W
07/08	78	x	LATIN SMALL LETTER X
07/09	79	y	LATIN SMALL LETTER Y
07/10	7A	z	LATIN SMALL LETTER Z
07/11	7B	{	LEFT CURLY BRACKET
07/12	7C	|	VERTICAL LINE
07/13	7D	}	RIGHT CURLY BRACKET
07/14	7E	~	TILDE

G.4 Reuter basic character set 2

The introduction to this appendix explains the notation used below.

Position		*Graphic*	*Name*
decimal	*hex*		
02/01	21	¡	INVERTED EXCLAMATION MARK
02/02	22	¢	CENT SIGN
02/03	23	£	POUND SIGN
02/04	24	R_T	REUTER RIGHTS SYMBOL
02/05	25	¥	YEN SIGN
02/06	26	W_I	REUTER WHEN ISSUED SYMBOL
02/07	27	§	SECTION SIGN
02/08	28	¤	CURRENCY SIGN
02/09	29	©	COPYRIGHT SIGN
02/10	2A	ª	FEMININE ORDINAL INDICATOR
02/11	2B	«	LEFT-POINTING DOUBLE ANGLE QUOTATION MARK
02/12	2C	⅛	VULGAR FRACTION ONE EIGHTH
02/13	2D	⅜	VULGAR FRACTION THREE EIGHTHS
02/14	2E	⅝	VULGAR FRACTION FIVE EIGHTHS
02/15	2F	⅞	VULGAR FRACTION SEVEN EIGHTHS
03/00	30	°	DEGREE SIGN
03/01	31	±	PLUS-MINUS SIGN
03/02	32	^{2}	SUPERSCRIPT TWO
03/03	33	^{3}	SUPERSCRIPT THREE
03/04	34	W_S	REUTER WARRANTS SYMBOL
03/05	35	µ	MICRO SIGN
03/06	36	¶	PILCROW SIGN
03/07	37		MIDDLE DOT
03/08	38	🔔	REUTER GRAPHIC BELL
03/09	39	^{1}	SUPERSCRIPT ONE
03/10	3A	º	MASCULINE ORDINAL INDICATOR
03/11	3B	»	RIGHT-POINTING DOUBLE ANGLE QUOTATION MARK
03/12	3C	¼	VULGAR FRACTION ONE QUARTER
03/13	3D	½	VULGAR FRACTION ONE HALF
03/14	3E	¾	VULGAR FRACTION THREE QUARTERS
03/15	3F	¿	INVERTED QUESTION MARK

Position		*Graphic*	*Name*
decimal	*hex*		
04/00	40	À	LATIN CAPITAL LETTER A WITH GRAVE
04/01	41	Á	LATIN CAPITAL LETTER A WITH ACUTE
04/02	42	Â	LATIN CAPITAL LETTER A WITH CIRCUMFLEX
04/03	43	Ã	LATIN CAPITAL LETTER A WITH TILDE
04/04	44	Ä	LATIN CAPITAL LETTER A WITH DIAERESIS
04/05	45	Å	LATIN CAPITAL LETTER A WITH RING ABOVE
04/06	46	Æ	LATIN CAPITAL LIGATURE AE
04/07	47	Ç	LATIN CAPITAL LETTER C WITH CEDILLA
04/08	48	È	LATIN CAPITAL LETTER E WITH GRAVE
04/09	49	É	LATIN CAPITAL LETTER E WITH ACUTE
04/10	4A	Ê	LATIN CAPITAL LETTER E WITH CIRCUMFLEX
04/11	4B	Ë	LATIN CAPITAL LETTER E WITH DIAERESIS
04/12	4C	Ì	LATIN CAPITAL LETTER I WITH GRAVE
04/13	4D	Í	LATIN CAPITAL LETTER I WITH ACUTE
04/14	4E	Î	LATIN CAPITAL LETTER I WITH CIRCUMFLEX
04/15	4F	Ï	LATIN CAPITAL LETTER I WITH DIAERESIS
05/00	50	P_R	REUTER PREFERRED SYMBOL
05/01	51	Ñ	LATIN CAPITAL LETTER N WITH TILDE
05/02	52	Ò	LATIN CAPITAL LETTER O WITH GRAVE
05/03	53	Ó	LATIN CAPITAL LETTER O WITH ACUTE
05/04	54	Ô	LATIN CAPITAL LETTER O WITH CIRCUMFLEX
05/05	55	Õ	LATIN CAPITAL LETTER O WITH TILDE
05/06	56	Ö	LATIN CAPITAL LETTER O WITH DIAERESIS
05/07	57	Œ	LATIN CAPITAL LIGATURE OE
05/08	58	Ø	LATIN CAPITAL LETTER O WITH STROKE
05/09	59	Ù	LATIN CAPITAL LETTER U WITH GRAVE
05/10	5A	Ú	LATIN CAPITAL LETTER U WITH ACUTE
05/11	5B	Û	LATIN CAPITAL LETTER U WITH CIRCUMFLEX
05/12	5C	Ü	LATIN CAPITAL LETTER U WITH DIAERESIS
05/13	5D	Ÿ	LATIN CAPITAL LETTER Y WITH DIAERESIS
05/14	5E	↑	UPWARDS ARROW
05/15	5F	β	LATIN SMALL LETTER SHARP S

Position		*Graphic*	*Name*
decimal	*hex*		
06/00	60	à	LATIN SMALL LETTER A WITH GRAVE
06/01	61	á	LATIN SMALL LETTER A WITH ACUTE
06/02	62	â	LATIN SMALL LETTER A WITH CIRCUMFLEX
06/03	63	ã	LATIN SMALL LETTER A WITH TILDE
06/04	64	ä	LATIN SMALL LETTER A WITH DIAERESIS
06/05	65	å	LATIN SMALL LETTER A WITH RING ABOVE
06/06	66	æ	LATIN SMALL LIGATURE AE
06/07	67	ç	LATIN SMALL LETTER C WITH CEDILLA
06/08	68	è	LATIN SMALL LETTER E WITH GRAVE
06/09	69	é	LATIN SMALL LETTER E WITH ACUTE
06/10	6A	ê	LATIN SMALL LETTER E WITH CIRCUMFLEX
06/11	6B	ë	LATIN SMALL LETTER E WITH DIAERESIS
06/12	6C	ì	LATIN SMALL LETTER I WITH GRAVE
06/13	6D	í	LATIN SMALL LETTER I WITH ACUTE
06/14	6E	î	LATIN SMALL LETTER I WITH CIRCUMFLEX
06/15	6F	ï	LATIN SMALL LETTER I WITH DIAERESIS
07/00	70	$\mathrm{U_N}$	REUTER UNITS SYMBOL
07/01	71	ñ	LATIN SMALL LETTER N WITH TILDE
07/02	72	ò	LATIN SMALL LETTER O WITH GRAVE
07/03	73	ó	LATIN SMALL LETTER O WITH ACUTE
07/04	74	ô	LATIN SMALL LETTER O WITH CIRCUMFLEX
07/05	75	õ	LATIN SMALL LETTER O WITH TILDE
07/06	76	ö	LATIN SMALL LETTER O WITH DIAERESIS
07/07	77	œ	LATIN SMALL LIGATURE OE
07/08	78	ø	LATIN SMALL LETTER O WITH STROKE
07/09	79	ù	LATIN SMALL LETTER U WITH GRAVE
07/10	7A	ú	LATIN SMALL LETTER U WITH ACUTE
07/11	7B	û	LATIN SMALL LETTER U WITH CIRCUMFLEX
07/12	7C	ü	LATIN SMALL LETTER U WITH DIAERESIS
07/13	7D	ÿ	LATIN SMALL LETTER Y WITH DIAERESIS
07/14	7E	↓	DOWNWARDS ARROW

H Restrictions on producing applications

As some existing consuming applications cannot handle the full range of mechanisms specified by this Standard, certain restrictions (described in this appendix) have been placed upon producing applications. Some of these restrictions may be lifted at a future time. It follows that designers of consuming applications should disregard these restrictions, so that their applications continue to operate if and when the restrictions no longer apply.

H.1 Shift functions

Some of the shift functions listed in figures 2.7 and 2.8 are reserved for future enhancement and must not be used at present. The permitted shift functions are shown in figure H.1, while shift functions which are not currently permitted are shown in figure H.2.

Shift function	*Acronym*
LOCKING-SHIFT ZERO	LS0
LOCKING-SHIFT ONE RIGHT	LS1R
LOCKING-SHIFT TWO RIGHT	LS2R
LOCKING-SHIFT THREE	LS3
SINGLE-SHIFT TWO	SS2
SINGLE-SHIFT THREE	SS3

Figure H.1 – Permitted shift functions

Shift function	*Acronym*
LOCKING-SHIFT ONE	LS1
LOCKING-SHIFT TWO	LS2
LOCKING-SHIFT THREE RIGHT	LS3R

Figure H.2 – Shift functions not currently permitted

H.2 Designation functions

Some of the designation functions listed in appendix D are reserved for future enhancement and must not be used at present. The permitted designation functions are shown in figure H.3. Forbidden designation functions are any which are *not* listed in that figure.

Character set	*Working set*	*Designation function*
Reuter basic character set 1	G0	ESC 02/08 04/02
Reuter basic character set 2	G1	ESC 02/09 03/01
Japanese Katakana character set	G2	ESC 02/10 03/02
Japanese Latin character set	G3	ESC 02/11 03/03
Japanese Kanji character set	G3	ESC 02/04 02/11 03/04
Chinese character set 1	G2	ESC 02/04 02/10 03/05
Chinese character set 2	G3	ESC 02/04 02/11 03/06

Figure H.3 – Permitted designation functions

H.3 NULLs

This Standard instructs the designers of consuming applications on the handling of trailing NULLs inserted for padding purposes by producing applications. Nevertheless, this use of NULLs is strongly discouraged and is to be avoided by the designers of new producing applications.

I Example

This appendix contains an example of a byte stream made up of byte sequences representing characters from various character sets, and constituting a single textual field. The upper row in each table contains the characters and the lower row contains the byte sequences representing those characters.

——————— Characters from Reuter basic character set 1 ———————

c	a	t		s	a	t		o	n		a		m	a	t
63	61	74	20	73	61	74	20	6F	6E	20	61	20	6D	61	74

——————— Characters from Reuter basic character sets 1 and 2 ———————

a	b	c	à	á	â	ã	ä	å	æ	ç	R_T	¥	W_I	¼	🔔
61	62	63	E0	E1	E2	E3	E4	E5	E6	E7	A4	A5	A6	BC	B8

—— The first seven Kanji characters[1] from the Japanese Kanji character set ——

LS3[2]		Kanji char 1		Kanji char 2		Kanji char 3		Kanji char 4		Kanji char 5		Kanji char 6		Kanji char 7	
1B	6F	30	21	30	22	30	23	30	24	30	25	30	26	30	27

—— The first five frequently-used Chinese characters[3] ——
from the Chinese character set 1

Designation function[4]				LS2R[5]		Chinese char 1		Chinese char 2		Chinese char 3		Chinese char 4		Chinese char 5	
1B	24	2A	35	1B	7D	C4	A1	C4	A2	C4	A3	C4	A4	C4	A5

——————————— Miscellaneous ———————————

LS1R[6]		à	á	â	SS2[7]	Chinese char 1[8]		ã	SS2[7]	Chinese char 3[8]		ä	SS2[7]	Chinese char 5[8]	
1B	7E	E0	E1	E2	8E	44	21	E3	8E	44	23	E4	8E	44	25

Notes

1 These are not the first seven characters of the character set, but the first seven Kanji characters. This character set also contains some other groups of characters.

2 LS3 invokes working set G3 to area GL.

3 These are not the first five characters of the character set, but the first five frequently-used Chinese characters. This character set also contains some other groups of characters.

4 This designation function designates Chinese character set 1 to working set G2.

5 LS2R invokes working set G2 to area GR.

6 LS1R invokes working set G1 to area GR.

7 SS2 invokes a single character from working set G2 to area GL.

8 Note that these byte values are lower by 80 than the byte values used immediately above to construct byte sequences representing the same characters.