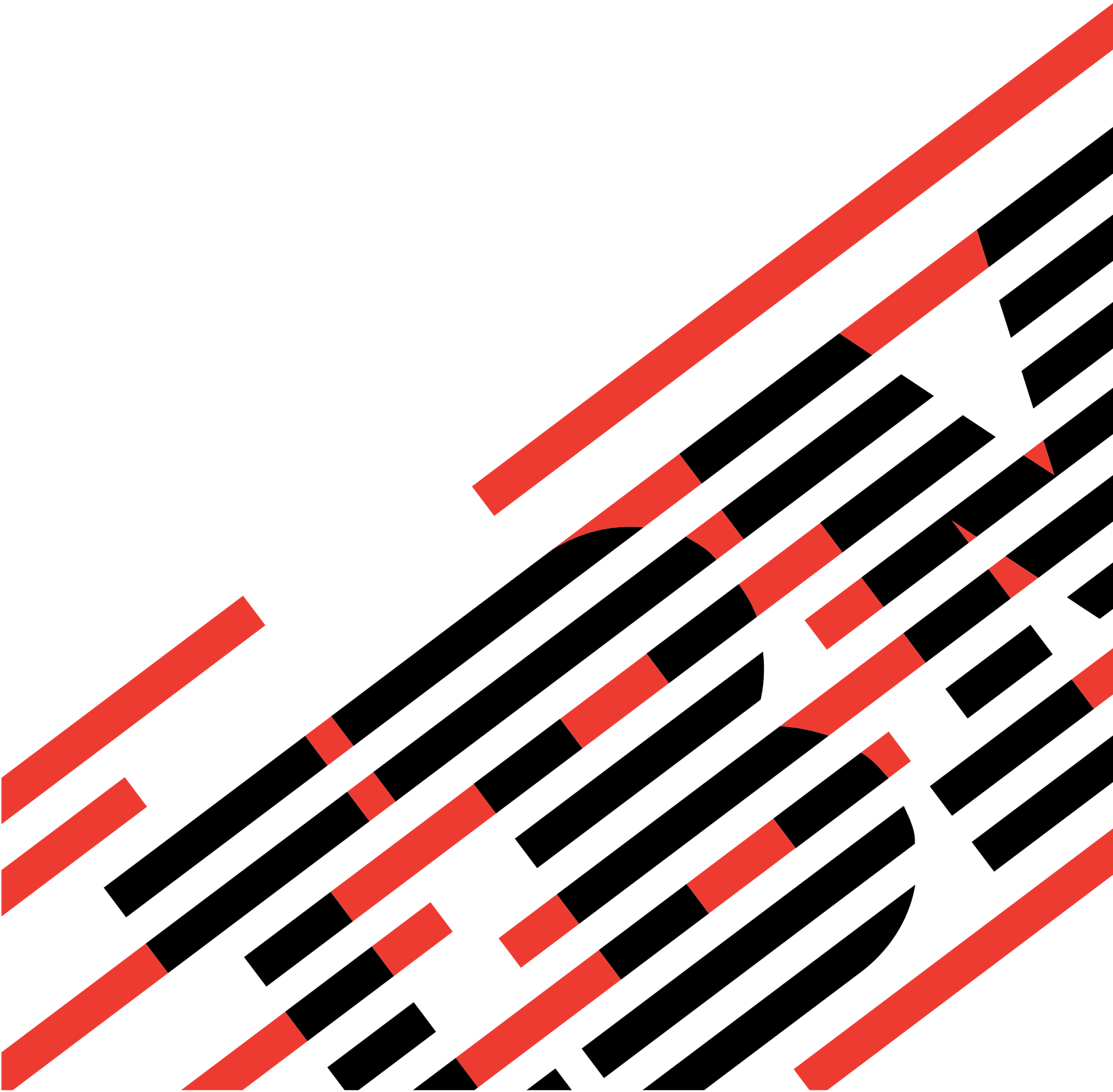
IBM

eserver

iSeries

Globalization (Handle data in global applications)



IBM

@server

iSeries

Globalization (Handle data in global applications)

© Copyright International Business Machines Corporation 1998, 2002. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 1998, 2002

OS/400 globalization

As companies integrate e-commerce on a global scale into their fundamental business processes, their prospective customers, established customers, and active partners can take advantage of increased revenue and decreased expenses through software globalization. They also can improve customer communications and increase savings. Globalized software gives you the following advantages:

- Increased customer satisfaction that can increase sales
- Enhanced customer support communications
- Enhanced global information dissemination
- A better return on Information Technology (IT) investments

This information shows you how to:

- Create an application efficiently and at minimal expense.
- Retrofit existing applications for globalization and create new applications designed for globalization. Designing an application for globalization, however, is usually less expensive than retrofitting an existing application.
- Ensure that the application design does not interfere with the current or planned design of other internationalized applications.

These pages provide a single source for the information you need to build applications for national and international audiences. You can also find information about what's new in this release and how you can print this topic.

Globalization overview
This topic describes the way that globalization has been implemented on OS/400(R), including topics that describe globalization-specific values on the system and topics that describe how services and functions in OS/400 support globalization.

Set up OS/400 with a national language version
This topic describes the steps you need to take to properly install and configure a national language version on OS/400, with topics ranging from selecting and installing hardware, installing software, and configuring your environment to run in a globalized setting. You can use this information as you install your own servers, and you can apply the principles when you develop applications for customers who are installing their own national language version on OS/400.

Develop global applications
This topic provides guidelines for designing, developing, and delivering globalized applications:

- Designing functions that are sensitive to national languages
- Supporting various types of hardware support
- Translating the textual data in your application
- Making your application available worldwide

Handle data in globalized applications
This topic describes the ways in which OS/400 enables you to handle data in a globalized environment. Included in these pages are topics that describe Unicode and UCS-2 data, the Chinese standard GB18030, how to use CCSIDs to integrate multiple language environments consistently, and how to use bidirectional data, DBCS data, and locales.

Globalization reference information
This topic provides detailed supporting information for the concepts and tasks discussed in the Globalization category.

© Copyright IBM Corp. 1998, 2002

Globalization checklists
This topic gathers together all of the checklists that are contained with these pages. These checklists are useful reminders of the issues you need to consider as you create and work with global applications.

Handle data in globalized applications

One of the most critical challenges you will face as you work with globalized servers and applications is the effective interaction with data. OS/400 provides a wide range of options that you can use to insure that data is viewed and processed seamlessly across national languages. The following topics describe globalization as it affects how you handle your data:

- Work with Unicode and UCS-2 data
- 
  GB18030: The Chinese standard
  
- Work with CCSIDs
- Work with bidirectional data
- Work with DBCS data
- Work with locales

Work with Unicode and UCS-2 data

Unicode

Prior to Unicode, the encoding systems that existed did not cover all the necessary numbers, characters, and symbols in use. Different encoding systems might assign the same number to different characters. If you used the wrong encoding system, your output might not have been what you expected to see.

Unicode provides a unique number for every character, regardless of platform, language, or program. Using Unicode, you can develop a software product that will work with various platforms, languages, and countries. Unicode also allows data to be transported through many different systems.

The following topics describe the Unicode implementation on OS/400:

- UTF-8
- 
  UTF-16
  
- 
  UTF-32
  
- How Unicode relates to prior standards such as ASCII and EBCDIC
- International components for Unicode (ICU)

For more information about Unicode, see the Unicode

web page.

UTF-8: A Unicode transformation format (UTF) is the algorithmic mapping from every Unicode value to a unique byte sequence. UTF-8 converts (via an algorithm) Unicode data so that it:

- No longer have nulls in it
- Uses 8 data bits to encode the data
- Keeps all ASCII codes from 00 to 7F as encoded as themselves

For example, the string ″ABC″ in Unicode would be ″004100420043″x. However, in UTF-8 it would be ″414243″.

Since UTF-8 allows Unicode data to flow over an 8-bit network without the network needing to know that it is Unicode, UTF-8 is used to store Unicode on several UNIX platforms and is used as the default encoding for most new internet standards.

OS/400 supports UTF-8 encoding with CCSID 1208.

UTF-16:

UTF-16 is an encoding of Unicode in which each character is composed of either one or two 16-bit elements.

OS/400 supports UTF-16 encoding with CCSID 1200.

Unicode was originally designed as a pure 16-bit encoding, aimed at representing all modern scripts. Over time, and especially after the addition of over 14,500 composite characters for compatibility with legacy sets, it became clear that 16 bits were not sufficient for most users. Out of this arose UTF-16.

UTF-16 allows access to about 60,000 characters as single Unicode 16-bit units. It can access an additional 1,000,000 characters by a mechanism known as surrogate pairs.

Two ranges of Unicode code values are reserved for the high (first) and low (second) values of these pairs. Highs are from 0xD800 to 0xDBFF, and lows from 0xDC00 to 0xDFFF. Since the most common characters have already been encoded in the first 64,000 values, the characters requiring surrogate pairs are relatively rare.

For more information about UTF-16, see the Unicode

web page.

UTF-32:

UTF-32 is an encoding of Unicode in which each character is composed of 4 bytes.

OS/400 does not support UTF-32 encoding with a CCSID value.

Unicode was originally designed as a pure 16-bit encoding, aimed at representing all modern scripts. Over time, and especially after the addition of over 14,500 composite characters for compatibility with legacy sets, it became clear that 16 bits were not sufficient for many users. Out of this arose UTF-32.

UTF-32 allows characters to be encoded as 4 bytes at any code point from 00000000 to 0010FFFF. For example, the string *ABC* in UTF-32 would be encoded as `x″00000041000000420000004300000043″`.

For more information about UTF-32, see the Unicode

web page.

How Unicode relates to prior standards such as ASCII and EBCDIC:

This topic provides a historical perspective on the Unicode standard, and explains how it can reduce the complexity of handling character data in global applications.

Evolving standards based on limited platforms

The representation of character data in modern computer systems can be fairly complicated, depending on the needs of your global application. One of the reasons for this complexity is that the methods for handling this data have evolved from early methods that served less complicated environments and hardware platforms.

In fact, many early decisions about how to encode characters on a system were guided by the functional requirements of specific devices, such as the early Telex (TTY) terminals and punch card technologies. For example, the Delete character (with an ASCII value of x'7F') was required in order to punch out all of the holes in a column of a punch card to signify that the column should be ignored. The storage capacities of these early computing systems placed additional limitations on system and application designers.

The character encoding schemes that have grown out of these early systems were built upon this historical foundation:

- The ASCII (American Standard Code for Information Interchange) character set uses 7-bit units, with a trivial encoding designed for 7-bit bytes. It is the most important character set in use today, despite its limitation to very few characters, because its design is the foundation for most modern character sets. ASCII provides only 128 numeric values, and 33 of those are reserved for special functions.
- The EBCDIC (Extended Binary-Coded Decimal Interchange Code) character set and a number of associated character sets, designed by IBM for its mainframes, uses 8-bit bytes. It was developed at a similar time as ASCII, and shares the same set of base characters and has other similar properties. Unlike ASCII, the Latin letters are not combined in two blocks for upper- and lower-case. Instead, the letters are arranged so that their hexadecimal values have second digits of 1 through 9 (another punch card-friendly design).

Historical simplicity creates modern complexity

The physical and functional limitations of the early character sets gave way to rapidly expanding hardware and functional capabilities. Character representation on computing systems became less dependent on hardware; instead, software designers used the existing encoding schemes to accommodate the needs of an increasingly global community of computer users.

Character sets for many characters

The most common encodings (character encoding schemes) use a single byte per character, and they are often called single-byte character sets (SBCS). They are all limited to 256 characters. Because of this, none of them can even cover all of the accented letters for the Western European languages. Consequently, many different such encodings were created over time to fulfill the needs of different user communities. The most widely used SBCS encoding today, after ASCII, is ISO-8859-1. It is an 8-bit superset of ASCII and provides most of the characters necessary for Western Europe.

However, East Asian writing systems needed a way to store over 10,000 characters, and so double-byte character sets (DBCS) were developed to provide enough space for the thousands of ideographic characters in East Asian writing systems. Here, the encoding is still byte-based, but each two bytes together represent a single character.

Even in East Asia, text contains letters from small alphabets like Latin or Katakana. These are represented more efficiently with single bytes. Multi-byte character sets (MBCS) provide for this by using a variable number of bytes per character, which distinguishes them from the DBCS encodings. MBCSs are often compatible with ASCII; that is, the Latin letters are represented in such encodings with the same bytes that ASCII uses. Some less often used characters may be encoded using three or even four bytes.

An important feature of MBCSs is that they have byte value ranges that are dedicated for lead bytes and trail bytes. Special ranges for lead bytes, the first bytes in multibyte sequences, make it possible to decide how many bytes belong together to encode a single character. Traditional MBCS encodings are designed so that it is easy to go forwards through a stream of bytes and read characters. However, it is often complicated and very dependent on the properties of the encoding to go backwards in text: going backwards, it is often hard to find out which variable number of bytes represents a single character, and sometimes it is necessary to go forward from the beginning of the text to do this.

Examples of commonly used MBCS encodings are Shift-JIS and EUC-JP (for Japanese), with up to two and three bytes per character, respectively.

Stateful encodings

Some encodings are stateful; they have bytes or byte sequences that switch the meanings of the following bytes. Simple encodings, like mixed-byte EBCDIC, use Shift-In and Shift-Out control characters (bytes) to switch between two states. Sometimes, the bytes after a Shift-In are interpreted as a certain SBCS encoding, and the bytes after a Shift-Out as a certain DBCS encoding. This is very different from an MBCS encoding where the bytes for each character indicate the length of the byte sequence.

The most common stateful encoding is ISO 2022 and its language-specific variations. It uses Escape sequences (byte sequences starting with an ASCII Escape character, byte value 27) to switch between many different embedded encodings. It can also *announce* encodings that are to be used with special shifting characters in the embedded byte stream. Language-specific variants like ISO-2022-JP limit the set of embeddable encodings and specify only a small set of acceptable Escape sequences for them.

Such encodings are very powerful for data exchange but hard to use in an application. Their flexibility allows you to embed many other encodings, but direct use in programs and conversions to and from other encodings are complicated. For direct use, a program has to keep track not only of the current position in the text, but also of the state—which embeddable encoding is currently active—or must be able to determine the state for a position from considerable context. For conversions to other encodings, converting software may need to have mappings for many embeddable encodings, and for conversions from other encodings, special code must figure out which embeddable encoding to choose for each character.

Unicode: The last character set?

The Unicode standard specifies a character set and several encodings. As of early 2002, it contains almost 94000 characters, which include all the characters of the common character sets that were in use

when Unicode was started around 1990, plus many that have been added since. It is an open character set, which means that it keeps growing and adding less frequently used characters.

The standard assigns numbers from 0 to 0x10FFFF, which is more than a million possible numbers for characters. About 5% of this space is used. Another 5% is in preparation, about 13% is reserved for private use (anyone can place any character in there), and about 2% is reserved and not to be used for characters. The remaining 75% is open for future use but not by any means expected to be filled up. In other words, there is finally a character set with plenty of space!

Unicode is in use today, and it is the preferred character set for the Internet, especially for HTML and XML. It is slowly being adopted for use in e-mail, too. Its most attractive property is that it covers all the characters of the world (with exceptions, which will be added in the future). Unicode makes it possible to access and manipulate characters by unique numbers—their Unicode code points—and use older encodings only for input and output, if at all.

Why Unicode?

Hundreds of encodings have been developed, each for small groups of languages and special purposes. As a result, the interpretation of text, input, sorting, display, and storage depends on the knowledge of all the different types of character sets and their encodings. Programs are written to either handle one single encoding at a time and switch between them, or to convert between external and internal encodings.

Part of the problem is that there is no single, authoritative source of precise definitions of many of the encodings and their names. Transferring of text from one machine to another one often causes some loss of information. Also, if a program has the code and the data to perform conversion between a significant subset of traditional encodings, then it carries several megabytes of data around.

Unicode provides a single character set that covers the languages of the world, and a small number of machine-friendly encoding forms and schemes to fit the needs of existing applications and protocols. It is designed for best interoperability with both ASCII and ISO-8859-1, the most widely used character sets, to make it easier for Unicode to be used in applications and protocols.

Unicode encodings

For single characters, 32-bit integer variables are most appropriate for the value range of Unicode. For strings, however, storing 32 bits for each character takes up too much space, especially considering that the highest value, 0x10FFFF, takes up only 21 bits. 11 bits are always unused in a 32-bit word storing a Unicode code point. Therefore, you will find that software generally uses 16-bit or 8-bit units as a compromise, with a variable number of code units per Unicode code point. It is a trade-off between ease of programming and storage space.

As a result, there are three common ways to store Unicode strings:

- UTF-32, with 32-bit code units, each storing a single code point
- UTF-16, with one or two 16-bit code units for each code point
- UTF-8, with one to four 8-bit code units (bytes) for each code point

UTF-8 is used mainly as a direct replacement for older MBCS encodings which all use 8-bit code units, but it takes some more code to process it. It is a good encoding if 90% of your data is English, since all English letters use only one byte.

UTF-16 is extremely well designed as the best compromise between handling and space, and all commonly used characters can be stored with one code unit per code point. This is the default encoding for Unicode.

International Components for Unicode: The International Components for Unicode (ICU) is a C library that provides a full-featured, industrial strength, Unicode support. The library provides:

- Calendar support
- Character set conversions
- Collation (language-sensitive)
- Date and time formatting
- Locales (140+ supported)
- Message catalogs (resources)
- Message formatting
- Normalization
- Number and currency formatting
- Time zones
- Transliteration
- Word, line, and sentence breaks

ICU is a collaborative, open-source development project jointly managed by a group of companies and individual volunteers throughout the world, using the Internet and the Web to communicate, plan, and develop the software and documentation.

The ICU project is licensed under the IBM Public License, which has been approved by the Open Source Initiative. For more information, see International Components for Unicode

(http://oss.software.ibm.com/icu/).

UCS-2 and its relationship to Unicode

Since the UCS-2 standard is limited to 65,535 characters, and the data processing industry needs over 94,000 characters, the UCS-2 standard is in the process of being superseded by the Unicode UTF-16 standard.

However, because UTF-16 is a superset of the existing UCS-2 standard, you can develop your applications using the systems existing UCS-2 support as long as your applications treat the UCS-2 as if it were UTF-16.

UCS (Universal Multiple-Octet Coded Character Set)

The ISO 10646 standard is a character code designed to encode text for storage in computer files. The design of the ISO 10646 standard is based on today's prevalent character code, ASCII (and ISO 8859-1, an extended version of the ASCII code). But ISO 10646 goes beyond ASCII's ability to encode only the Latin alphabet. The ISO 10646 encoding provides the capability to encode all of the characters used for written languages throughout the world.

Two UCS encoding schemes

In order to accommodate the many thousands of characters used in international text, ISO/IEC 10646 specifies the Universal Multiple-Octet Coded Character Set (UCS). UCS can be implemented through two encoding schemes:

- UCS-2: Each character is represented by 16 bits or 2 bytes. (The number 2 in UCS-2 indicates 2 bytes.) For example, uppercase A is represented by 0041.

- UCS-4: Each character is represented by 32 bits or 4 bytes. (The number 4 in UCS-4 indicates 4 bytes.) For example, uppercase A is represented by 0000 0041.

The major difference between the 2-byte and 4-byte representation is that the 4-byte representation allows for the presentation or use of additional characters beyond the capability of UCS-2. That is, you can encode more characters in UCS-4 than you can in UCS-2.

Benefits of UCS over ASCII

UCS provides codes for more than 65,000 characters, a huge increase over ASCII's 7-bit code capacity of 128 characters. To keep character coding simple and efficient, the UCS-2 standard assigns each character a unique 16-bit value, and does not use complex modes or escape codes to specify modified characters or special cases. This simplicity and efficiency makes it easy for computers and software to handle ISO 10646-encoded text files.

UCS-2 allows for the use of ″combining characters″. A combining character is a non-spacing character that is used together with a non-combining character to form a composite character, or glyph. For example, Latin small letter A used with a combining tilde results in

ã

.

Within UCS-2 and UCS-4, characters can be presented or used at various levels. The levels and their descriptions are:

- Level 1: No use of combining characters is allowed.
- Level 2: Limited use of combining characters is allowed.
- Level 3: No restriction on use of combining characters.

The following topics provide more detailed information about UCS-2 support on OS/400:

- Why use UCS-2?
- UCS-2 on OS/400
- UCS-2 level-1 mapping tables

Why use UCS-2?: OS/400 provides multilingual support. UCS-2 provides the means to store and retrieve data in the user's national language of choice in a single file and therefore provides for one database file to support all text needs, regardless of the language of the input device. For example, the same parts file could have Greek, Russian, and English descriptions and names in it.

Mapping of data

OS/400 uses the EBCDIC encoding scheme. However, not all clients attached to it use an EBCDIC encoding scheme to store, retrieve, and process data. For example, some clients may use ASCII, PC DATA, or other encoding schemes. Using UCS-2 prevents the loss of data due to incomplete conversion between encoding schemes and code pages. Therefore, some clients use UCS-2 as an "exchange mechanism" that is safe across all platforms.

Examples:

The following examples highlight two users on the same system. One user is English and the other Greek. The English user has his display device CCSID set to 37. The Greek user has his display device CCSID set to 875. Both users query, update, and replace data in the DATABASE1. DATABASE1 is tagged with CCSID 37.

- Example 1: Displaying data without UCS-2
- Example 2: Displaying data with UCS-2

Example: Display data without UCS-2: Problems with data integrity develop because users are operating with CCSIDs that have varied character support. That is, not all characters in CCSID 37 are available in CCSID 875 and vice-versa.

Assume that the following names are to be entered by the English-speaking user (display device supports a CCSID of 37):

- Å
  alson
- Gifford

When these entries are stored, the data integrity remains intact. That is, an

Å

is stored as an

Å

. This is because the display device CCSID and the database CCSID are both 37.

Assume the following names are also input into DATABASE1 by the Greek-speaking user (display device CCSID of 875):

- Å π έ ν
- Ω ρ ι μ α

DATABASE1 now consists of the following logical entries:

- Å
  alson
- Gifford

- Μ
  π
  έ
  ν
- Ω
  ρ
  ι
  μ
  α

The Greek characters that make up the name are stored as those characters only if the same character exists within CCSID 37. If the character does not exist, the server converts the characters using a predetermined algorithm to a code point from code page 37. The algorithm converts

Ω

to

Å

.

The following list shows the code point used to store the first character of each name in DATABASE1. (Using only the first character makes the example easier by eliminating long strings of code points which would be shown if we presented the code point for each character in the name.)

Name CCSID 37 Stored Code Point (Hexadecimal)

Å

alson 67 . . .

Gifford
C7 . . .

Μ

π

έ

ν

53 . . .

Ω

ρ

ι

μ

α

67 . . .

The next step in this example is to show how data can be incorrectly selected due to the character conversion when it was stored in the database.

Assume the Greek user wants to find all names beginning with

Ω

. The following SQL statement would provide two names:

Ω

ρ

ι

μ

α

and

Å

alson

```
Select from DATABASE1 where name LIKE '%'
```

The search yielded an unexpected name (

Å

alson). This is because the first character in

Å

alson is stored with the same code point as the first character in

Ω

ρ

ι

μ

α

.

Example: Display data with UCS-2: In this example, using UCS-2 as the CCSID of DATABASE1, we can show how data integrity is maintained both in storing and retrieving data. As in the previous example, one user is English using CCSID 37 and the other user is Greek using CCSID 875.

We'll use DATABASE1 as in the previous example. However DATABASE1 is now defined with CCSID 13488. (13488 is a UCS-2 CCSID.)

- Å
  alson
- Gifford
- M

  π

  έ

  ν
- Ω

  ρ

  ι

  ι

  μ

  α

The key difference in using UCS-2 as the CCSID of DATABASE1 is that data integrity is maintained for each user who inputs data to the database. That is each character, regardless of the CCSID of the inputting device, is stored with a unique code point. (Remember that in this example the CCSID of DATABASE1 is 13488.)

Name CCSID 13488 Stored Code Point (Hexadecimal)

Å

alson 00C5 . . .

Gifford
0047 . . .

M

π

έ

03A9 . . .

Ω

ρ

ι

μ

α

039C . . .

Assume the Greek user wants to find all names beginning with

Ω

. The following SQL statement would provide one name,

Ω

Ω

ρ

ι

μ

α

, as compared to two in the previous example:

```
Select from DATABASE1 where Substr(name,1,1) = ''
```

The reason for this is that each character stored in a UCS-2 tagged database has a unique code point. This contrasts to the previous example that had the first character in

Å

also stored with the same code point as the first character in

Ω

ρ

ι

μ

α

UCS-2 on OS/400: OS/400 supports UCS-2 and implements its UCS-2 conversion support using level 1 support. That is, no use of combining characters is mapped.

The coded character set identifier (CCSID) 13488 on OS/400 represents UCS-2.

UCS-2 cannot be specified as a value for:

- The system CCSID
- A user profile CCSID
- A job CCSID

OS/400 provides external support for UCS-2 in the following parts of the system (see note, below):

- Database files and functions
- DB2 UDB for iSeries
- SQL tables
- Query files and tools
- DDS
- Display file and panel groups
- Sort sequences
- UCS-2 variables in UIM
- ILE high-level languages such as RPG
- Message handling and message catalogs

Several other OS/400 functions implement UCS-2 internally so that character data integrity is maintained for users across multilingual platforms.

Note: These topics do not give detailed information on application development as it relates to the implementation of UCS-2. Rather, they provide highlights of OS/400 support for UCS-2. Where possible, reference to a book that provides detailed information for UCS-2 implementation is given.

You should have available and understand the information in the Unicode standard.

For more information about Unicode, see the Unicode

web page.

Database files and functions: When you create UCS-2 database applications, you need to consider the implications for creating physical files (see page 15), creating logical files (see page 15), and performing input and output on the database files (see page 16).

Creating physical files:

UCS-2 graphic fields can be created in physical files. This is done by specifying a G data type and a UCS-2 CCSID for the CCSID keyword.

The following example shows the DDS for a physical file containing four fields, and the command for creating the file:

```
A          R FMT1
A            EMPNO          6A
A            NAME          30G         CCSID(13488)
A            DESCR1       500G         CCSID(13488) VARLEN
A            DESCR2       500A

CRTPF FILE(UCS-2PF) SRCFILE(CLR/QDDSSRC)
```

In the example:

- The first field, `EMPNO`, is a character field of length 6. The CCSID of the EMPNO field is the SBCS CCSID of the job. The decision was made to use a character field because the EMPNO field contains only numerics and UCS-2 support is not needed.
- The `NAME` and `DESCR1` fields are both UCS-2 fields. Both of these fields may need to contain data from more than one EBCDIC code page so the decision was made to make these fields UCS-2 graphic.
- The `DESCR2` field is the SBCS CCSID of the job. This field is used as illustratioin of mapping to a logical field in Creating logical files (see page 15).

You can specify the default (DFT) keyword for UCS-2 graphic fields. The default value can be specified as SBCS, bracketed-DBCS, or bracketed-DBCS-graphic character strings. If you do not specify the DFT keyword, the default value for fixed-length UCS-2 fields is the UCS-2 blank (hexadecimal 0020). For varying-length UCS-2 fields, the default is the empty string.

Creating logical files:

You can use logical files to map UCS-2 data to and from character, DBCS-open, or DBCS-graphic. This allows UCS-2 graphic data to be manipulated in a character based form.

The following example shows the DDS for a logical file containing 4 character fields. The UCS-2 graphic data is converted to character data when reading from the logical file, and character data is converted to UCS-2 graphic data when writing to the file.

```
R FMT1                      PFILE(UCS2PF1)
A            EMPNO
A            NAME            A         CCSID(37)
A            DESCR1          A         CCSID(37)
A            DESCR2          G         CCSID(13488)
```

Database input/output:

Whenever reading or writing data from or to a field tagged with a UCS-2 CCSID to the job physical files, the data is passed as UCS-2 data without any conversions occurring. Regardless of the job CCSID, data is passed as UCS-2 data. When writing data to a logical file, the *from* CCSID is the job CCSID; however, if the job CCSID is 65535, the *from* CCSID is the CCSID of the field in the logical file.

The following are some scenarios from the physical and logical files listed above. For the scenarios, the job CCSID is 297.

Scenario 1. When reading the data from the physical file:
- EMPNO is converted from its CCSID to 297.
- NAME is not converted but is left as UCS-2 data.
- DESCR1 is not converted but is left as UCS-2 data.
- DESCR2 is converted from its CCSID to 297.

Scenario 2. When writing the data to the physical file:
- EMPNO is converted from 297 to its CCSID.
- NAME is not converted but is left as UCS-2 data.
- DESCR1 is not converted but is left as UCS-2 data.
- DESCR2 is converted from 297 to its CCSID.

Scenario 3. When reading the data from the logical file:
- EMPNO is converted from its CCSID to 297.
- NAME is converted from UCS-2 data to character data with a CCSID of 297.
- DESCR1 is converted from UCS-2 data to character data with a CCSID of 297.
- DESCR2 is converted from character data to UCS-2 data and not converted to the job CCSID.

Scenario 4. When writing the data to the logical file:
- EMPNO is converted from 297 to its CCSID.
- NAME is converted from 297 to UCS-2 data.
- DESCR1 is converted from 297 to UCS-2 data.
- DESCR2 is converted from UCS-2to its CCSID in the physical file.

Scenario 5. If the job was 65535, the conversions for the above fields would be:
- EMPNO is not converted.
- NAME is converted from 37 to UCS-2 data.
- DESCR1 is converted from 37 to UCS-2 data.
- DESCR2 is converted from UCS-2 to its CCSID in the physical file.

DB2 UDB for iSeries: Keep the following in mind when using DB2 UDB for iSeries applications:
- Implicit conversion when comparing UCS-2 fields with character/IGC/graphic fields as well as with literals and host variables can occur.
- Physical and logical files with UCS-2 fields cannot have their CCSIDs changed with the Change Physical File (CHGPF) command.
- A UCS-2 CCSID is not allowed on the CHGPF command.
- The Copy File (CPYF) and Copy From Query File (CPYFRMQRYF) commands with FMTOPT(*MAP) specified is not allowed when copying from or to a UCS-2 graphic field unless:
  - the corresponding field is a UCS-2 or DBCS-graphic field.

  - the corresponding field is a character, DBCS-open, DBCS-either, or DBCS-only field with a CCSID other than 65535.
- The Copy File (CPYF) command supports copying of SBCS character, DBCS-open, DBCS-only, DBCS-either, and DBCS-graphic fields to and from UCS-2 graphic fields. There is limited support for UCS-2 on the FROMKEY, TOKEY, INCCHAR, and INCREL parameters.

SQL tables: SQL supports tables that contain UCS-2 graphic columns by specifying a UCS-2 CCSID for the GRAPHIC and VARGRAPHIC data types.

The following SQL example creates the table UCS2_TABLE. UCS2_TABLE contains one character column called EMPNO, and two UCS-2 graphic columns. NAME is a fixed-length UCS-2 graphic column and DESCRIPTION is a variable-length UCS-2 graphic column. The decision was made to use a character field since the EMPNO field only contains numerics and UCS-2 support is not needed. The NAME and DESCRIPTION fields are both UCS-2 fields. Both of these fields may contain data from more than one EBCDIC code page.

```
CREATE TABLE UCS2_TABLE (EMPNO CHAR(6) NOT NULL,
NAME GRAPHIC(30) CCSID 13488,
DESCRIPTION VARGRAPHIC(500) CCSID 13488)
```

Inserting data

SBCS character, mixed character, and DBCS graphic data can be inserted into UCS-2 graphic columns using the SQL INSERT statement. DB2 UDB for iSeries SQL converts the data to UCS-2 graphic. In SQL programs, the DECLARE VARIABLE statement can be used to attach a UCS-2 CCSID to graphic host variables.

The following SQL example converts character data to UCS-2 graphic for the NAME and DESCRIPTION columns and inserts the row into the UCS2_TABLE.

```
INSERT INTO UCS2_TABLE VALUES('000001','John Doe','Engineer')
```

Selecting UCS-2 data

Implicit conversion of UCS-2 graphic data is supported on a FETCH or select INTO and CALL.

In the following example, the EMPNO column is returned in empno_hv as character data. The NAME column is returned in name_hv as UCS-2 graphic data because name_hv is a UCS-2 variable. It is not converted to character, mixed character, or DBCS graphic.

```
...
char empno_hv[7];
wchar_t name_hv[31];
EXEC SQL DECLARE :name_hv VARIABLE CCSID 13488;
...
EXEC SQL SELECT EMPNO, NAME
INTO :empno_hv, :name_hv
.FROM UCS2_TABLE;
...
```

To return UCS-2 graphic data as EBCDIC data, the prior example could be changed to return the UCS-2 data as character data, EMPNO and NAME are returned in the job CCSID.

```
...
char empno_hv[7];
char name_hv[31];
...
EXEC SQL SELECT EMPNO, NAME
INTO :empno_hv, :name_hv
FROM UCS2_TABLE;
...
```

When doing selection, implicit conversions is done when comparing UCS-2 graphic data and character or DBCS graphic data.

The following example converts the character string 'John Doe' to UCS-2 graphic and then selects the rows where the NAME column is 'John Doe'.

```
EXEC SQL DECLARE C1 CURSOR FOR
SELECT *
FROM UCS2_TABLE
WHERE NAME = 'John Doe';
```

For additional information on using SQL with UCS-2 graphic data, see the SQL Reference topic in the Information Center.

Query files and tools: **Open query file (OPNQRYF) command considerations**

The Open Query File (OPNQRYF) command, as shown below, can retrieve or perform selection of UCS-2 data. Using the MAPFLD parameter, data can be mapped to or from UCS-2.

```
OPNQRYF FILE(UCS2_TABLE)
QRYSLT('NAME=MAPNAME')
MAPFLD((MAPNAME 'John Doe' *GRAPHIC *N *N 13488))
```

Interactive query tools considerations

Query for iSeries, DB2 Query Manager, and the DB2 Query Management function for OS/400 all have UCS-2 support. UCS-2 data can be displayed or printed on a report; by implicitly converting to either character or mixed art.

For additional information, see the Query Manager Use

and Query Management Programming

PDFs.

Data description specifications (DDS): In DDS, you use the CCSID file-, record-, or field-level keyword to specify that a G-type field supports UCS-2 data instead of DBCS-graphical data. See the CCSID keyword description in the DDS Reference: Physical and Logical Files topic.

The following are DDS considerations for UCS-2 and OS/400 applications:

- UCS-2 CCSID 13488 can be specified for graphic and variable graphic fields in physical files. UCS-2 CCSID 61952 cannot be specified in physical files.
- Logical files can be used to map from UCS-2 fields in the physical file to character (A or O) or DBCS graphic in the logical. Logical files can also be used to map character (A or O) or DBCS graphic in the physical file to UCS-2 graphic in the logical file. A CCSID can be specified in a DDS logical file. If the CCSID parameter is specified, the logical file is created using that CCSID. If a CCSID is not specified, the job default CCSID is used if mapping from UCS-2 to character is specified.

  If a logical file is used for I/O, fields are defined as character or DBCS graphic and the underlying physical fields are defined as UCS-2. On output the data is mapped directly from the job CCSID to UCS-2. Data will not map first from the job CCSID to the logical file CCSID and then from the logical file CCSID to the physical file UCS-2 CCSID. This mapping prevents data loss. On input, the UCS-2 data is mapped directly to the job CCSID.

- If a UCS-2 CCSID is specified at the file level and there are character fields defined for the file, the file can be created and the job default CCSID is used for the fields that do not have an explicit CCSID specified.
- If the field has a UCS-2 CCSID and a user-specified default value is not specified, then the default is UCS-2 blanks (X'0020') for fixed-length UCS-2 graphic and the empty string for varying-length UCS-2 graphic. A user-specified default may be specified as either a character or graphic literal. This literal value is converted to UCS-2 by database and stored internally in UCS-2.

Display files and panel groups: UCS-2 data is not supported on display devices that currently support the 5250 data stream. Therefore, conversions between the UCS-2 data and EBCDIC are necessary during input/output operations. On output, the UCS-2 data is converted to the CCSID of the device. On input, the data is converted from the device CCSID to the UCS-2 CCSID.

Since the device CCSID, which is determined from the device configuration, determines what the UCS-2 data is converted to, the converted data appears differently on different devices. For example, a UCS-2 character which maps to a SBCS character is displayed as a DBCS replacement character on a graphic-DBCS capable device. On a DBCS or SBCS capable device, the character appears as a SBCS character. A UCS-2 character which maps to a DBCS character is displayed as a graphic-DBCS character on a graphic-DBCS capable device. On a DBCS device, a DBCS character is bracketed (enclosed in a shift-out and shift-in). A SBCS replacement character is displayed on a SBCS device.

It is also suggested that all UCS-2 capable fields are initialized in the output buffer before writing the fields to the screen. Unpredictable results may occur if default initialization is allowed to take place.

For more information about display file and panel group considerations, see the UCS-2 appendix in the DDS Reference: Physical and Logical topic.

UCS-2 variables in UIM: The following example shows how to define a UCS-2 variable in UIM.

```
1     :class name=example basetype='graphic 6 13488' width=10,
2
3     :class name=example2 basetype='graphic 10 13488' width=20.
4
```

Line 1 defines a class for variables that will contain 6 UCS-2 characters and is to be displayed in a field that is 10 bytes long.

Line 3 defines a class for variables that will contain 10 UCS-2 characters and is to be displayed in a field that is 20 bytes long.

For more information on UCS-2 and UIM, see the definition of the CLAS tag in the Application Display Programming

PDF.

UCS-2 level-1 mapping tables: You can convert characters encoded in universal coded character set 2 level 1 (UCS-2 level-1) from uppercase to lowercase. The Uppercase to lowercase mapping table shows the mapping for this conversion.

You can also convert UCS-2 level-1 characters from lowercase to uppercase. The Lowercase to uppercase mapping table shows the mapping for this conversion.

Use the Convert Case API to perform these conversions.

ISO 10646 uppercase to lowercase UCS-2 level-1 conversion mapping*:*

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
0041	0061	LATIN CAPITAL LETTER A	LATIN SMALL LETTER A
0042	0062	LATIN CAPITAL LETTER B	LATIN SMALL LETTER B
0043	0063	LATIN CAPITAL LETTER C	LATIN SMALL LETTER C
0044	0064	LATIN CAPITAL LETTER D	LATIN SMALL LETTER D
0045	0065	LATIN CAPITAL LETTER E	LATIN SMALL LETTER E
0046	0066	LATIN CAPITAL LETTER F	LATIN SMALL LETTER F
0047	0067	LATIN CAPITAL LETTER G	LATIN SMALL LETTER G
0048	0068	LATIN CAPITAL LETTER H	LATIN SMALL LETTER H
0049	0069	LATIN CAPITAL LETTER I	LATIN SMALL LETTER I
004A	006A	LATIN CAPITAL LETTER J	LATIN SMALL LETTER J
004B	006B	LATIN CAPITAL LETTER K	LATIN SMALL LETTER K
004C	006C	LATIN CAPITAL LETTER L	LATIN SMALL LETTER L
004D	006D	LATIN CAPITAL LETTER M	LATIN SMALL LETTER M
004E	006E	LATIN CAPITAL LETTER N	LATIN SMALL LETTER N
004F	006F	LATIN CAPITAL LETTER O	LATIN SMALL LETTER O
0050	0070	LATIN CAPITAL LETTER P	LATIN SMALL LETTER P
0051	0071	LATIN CAPITAL LETTER Q	LATIN SMALL LETTER Q
0052	0072	LATIN CAPITAL LETTER R	LATIN SMALL LETTER R
0053	0073	LATIN CAPITAL LETTER S	LATIN SMALL LETTER S
0054	0074	LATIN CAPITAL LETTER T	LATIN SMALL LETTER T
0055	0075	LATIN CAPITAL LETTER U	LATIN SMALL LETTER U
0056	0076	LATIN CAPITAL LETTER V	LATIN SMALL LETTER V
0057	0077	LATIN CAPITAL LETTER W	LATIN SMALL LETTER W
0058	0078	LATIN CAPITAL LETTER X	LATIN SMALL LETTER X
0059	0079	LATIN CAPITAL LETTER Y	LATIN SMALL LETTER Y
005A	007A	LATIN CAPITAL LETTER Z	LATIN SMALL LETTER Z
00C0	00E0	LATIN CAPITAL LETTER A GRAVE	LATIN SMALL LETTER A GRAVE
00C1	00E1	LATIN CAPITAL LETTER A ACUTE	LATIN SMALL LETTER A GRAVE
00C2	00E2	LATIN CAPITAL LETTER A CIRCUMFLEX	LATIN SMALL LETTER A GRAVE
00C3	00E3	LATIN CAPITAL LETTER A TILDE	LATIN SMALL LETTER A GRAVE
00C4	00E4	LATIN CAPITAL LETTER A DIAERESIS	LATIN SMALL LETTER A GRAVE
00C5	00E5	LATIN CAPITAL LETTER A RING	LATIN SMALL LETTER A GRAVE
00C6	00E6	LATIN CAPITAL LETTER A E	LATIN SMALL LETTER A GRAVE
00C7	00E7	LATIN CAPITAL LETTER C CEDILLA	LATIN SMALL LETTER A GRAVE

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
00C8	00E8	LATIN CAPITAL LETTER E GRAVE	LATIN SMALL LETTER A GRAVE
00C9	00E9	LATIN CAPITAL LETTER E ACUTE	LATIN SMALL LETTER A GRAVE
00CA	00EA	LATIN CAPITAL LETTER E CIRCUMFLEX	LATIN SMALL LETTER E CIRCUMFLEX
00CB	00EB	LATIN CAPITAL LETTER E DIAERESIS	LATIN SMALL LETTER E DIAERESIS
00CC	00EC	LATIN CAPITAL LETTER I GRAVE	LATIN SMALL LETTER I GRAVE
00CD	00ED	LATIN CAPITAL LETTER I ACUTE	LATIN SMALL LETTER I ACUTE
00CE	00EE	LATIN CAPITAL LETTER I CIRCUMFLEX	LATIN SMALL LETTER I CIRCUMFLEX
00CF	00EF	LATIN CAPITAL LETTER I DIAERESIS	LATIN SMALL LETTER I DIAERESIS
00D0	00F0	LATIN CAPITAL LETTER ETH	LATIN SMALL LETTER ETH
00D1	00F1	LATIN CAPITAL LETTER N TILDE	LATIN SMALL LETTER N TILDE
00D2	00F2	LATIN CAPITAL LETTER O GRAVE	LATIN SMALL LETTER O GRAVE
00D3	00F3	LATIN CAPITAL LETTER O ACUTE	LATIN SMALL LETTER O ACUTE
00D4	00F4	LATIN CAPITAL LETTER O CIRCUMFLEX	LATIN SMALL LETTER O CIRCUMFLEX
00D5	00F5	LATIN CAPITAL LETTER O TILDE	LATIN SMALL LETTER O TILDE
00D6	00F6	LATIN CAPITAL LETTER O DIAERESIS	LATIN SMALL LETTER O DIAERESIS
00D8	00F8	LATIN CAPITAL LETTER O SLASH	LATIN SMALL LETTER O SLASH
00D9	00F9	LATIN CAPITAL LETTER U GRAVE	LATIN SMALL LETTER U GRAVE
00DA	00FA	LATIN CAPITAL LETTER U ACUTE	LATIN SMALL LETTER U ACUTE
00DB	00FB	LATIN CAPITAL LETTER U CIRCUMFLEX	LATIN SMALL LETTER U CIRCUMFLEX
00DC	00FC	LATIN CAPITAL LETTER U DIAERESIS	LATIN SMALL LETTER U DIAERESIS
00DD	00FD	LATIN CAPITAL LETTER Y ACUTE	LATIN SMALL LETTER Y ACUTE
00DE	00FE	LATIN CAPITAL LETTER THORN	LATIN SMALL LETTER THORN
0100	0101	LATIN CAPITAL LETTER A WITH MACRON	LATIN SMALL LETTER A WITH MACRON
0102	0103	LATIN CAPITAL LETTER A WITH BREVE	LATIN SMALL LETTER A WITH BREVE

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
0104	0105	LATIN CAPITAL LETTER A WITH OGONEK	LATIN SMALL LETTER A WITH OGONEK
0106	0107	LATIN CAPITAL LETTER C WITH ACUTE	LATIN SMALL LETTER C WITH ACUTE
0108	0109	LATIN CAPITAL LETTER C WITH CIRCUMFLEX	LATIN SMALL LETTER C WITH CIRCUMFLEX
010A	010B	LATIN CAPITAL LETTER C WITH DOT ABOVE	LATIN SMALL LETTER C WITH DOT ABOVE
010C	010D	LATIN CAPITAL LETTER C WITH CARON	LATIN SMALL LETTER C WITH CARON
010E	010F	LATIN CAPITAL LETTER D WITH CARON	LATIN SMALL LETTER D WITH CARON
0110	0111	LATIN CAPITAL LETTER D WITH STROKE	LATIN SMALL LETTER D WITH STROKE
0112	0113	LATIN CAPITAL LETTER E WITH MACRON	LATIN SMALL LETTER E WITH MACRON
0114	0115	LATIN CAPITAL LETTER E WITH BREVE	LATIN SMALL LETTER E WITH BREVE
0116	0117	LATIN CAPITAL LETTER E WITH DOT ABOVE	LATIN SMALL LETTER E WITH DOT ABOVE
0118	0119	LATIN CAPITAL LETTER E WITH OGONEK	LATIN SMALL LETTER E WITH OGONEK
011A	011B	LATIN CAPITAL LETTER E WITH CARON	LATIN SMALL LETTER E WITH CARON
011C	011D	LATIN CAPITAL LETTER G WITH CIRCUMFLEX	LATIN SMALL LETTER G WITH CIRCUMFLEX
011E	011F	LATIN CAPITAL LETTER G WITH BREVE	LATIN SMALL LETTER G WITH BREVE
0120	0121	LATIN CAPITAL LETTER G WITH DOT ABOVE	LATIN SMALL LETTER G WITH DOT ABOVE
0122	0123	LATIN CAPITAL LETTER G WITH CEDILLA	LATIN SMALL LETTER G WITH CEDILLA
0124	0125	LATIN CAPITAL LETTER H WITH CIRCUMFLEX	LATIN SMALL LETTER H WITH CIRCUMFLEX
0126	0127	LATIN CAPITAL LETTER H WITH STROKE	LATIN SMALL LETTER H WITH STROKE
0128	0129	LATIN CAPITAL LETTER I WITH TILDE	LATIN SMALL LETTER I WITH TILDE
012A	012B	LATIN CAPITAL LETTER I WITH MACRON	LATIN SMALL LETTER I WITH MACRON
012C	012D	LATIN CAPITAL LETTER I WITH BREVE	LATIN SMALL LETTER I WITH BREVE
012E	012F	LATIN CAPITAL LETTER I WITH OGONEK	LATIN SMALL LETTER I WITH OGONEK
0130	0069	LATIN CAPITAL LETTER I WITH DOT ABOVE	LATIN SMALL LETTER I
0132	0133	LATIN CAPITAL LIGATURE IJ	LATIN SMALL LIGATURE IJ

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
0134	0135	LATIN CAPITAL LETTER J WITH CIRCUMFLEX	LATIN SMALL LETTER J WITH CIRCUMFLEX
0136	0137	LATIN CAPITAL LETTER K WITH CEDILLA	LATIN SMALL LETTER K WITH CEDILLA
0139	013A	LATIN CAPITAL LETTER L WITH ACUTE	LATIN SMALL LETTER L WITH ACUTE
013B	013C	LATIN CAPITAL LETTER L WITH CEDILLA	LATIN SMALL LETTER L WITH CEDILLA
013D	013E	LATIN CAPITAL LETTER L WITH CARON	LATIN SMALL LETTER L WITH CARON
013F	0140	LATIN CAPITAL LETTER L WITH MIDDLE DOT	LATIN SMALL LETTER L WITH MIDDLE DOT
0141	0142	LATIN CAPITAL LETTER L WITH STROKE	LATIN SMALL LETTER L WITH STROKE
0143	0144	LATIN CAPITAL LETTER N WITH ACUTE	LATIN SMALL LETTER N WITH ACUTE
0145	0146	LATIN CAPITAL LETTER N WITH CEDILLA	LATIN SMALL LETTER N WITH CEDILLA
0147	0148	LATIN CAPITAL LETTER N WITH CARON	LATIN SMALL LETTER N WITH CARON
014A	014B	LATIN CAPITAL LETTER ENG (SAMI)	LATIN SMALL LETTER ENG (SAMI)
014C	014D	LATIN CAPITAL LETTER O WITH MACRON	LATIN SMALL LETTER O WITH MACRON
014E	014F	LATIN CAPITAL LETTER O WITH BREVE	LATIN SMALL LETTER O WITH BREVE
0150	0151	LATIN CAPITAL LETTER O WITH DOUBLE ACUTE	LATIN SMALL LETTER O WITH DOUBLE ACUTE
0152	0153	LATIN CAPITAL LIGATURE OE	LATIN SMALL LIGATURE OE
0154	0155	LATIN CAPITAL LETTER R WITH ACUTE	LATIN SMALL LETTER R WITH ACUTE
0156	0157	LATIN CAPITAL LETTER R WITH CEDILLA	LATIN SMALL LETTER R WITH CEDILLA
0158	0159	LATIN CAPITAL LETTER R WITH CARON	LATIN SMALL LETTER R WITH CARON
015A	015B	LATIN CAPITAL LETTER S WITH ACUTE	LATIN SMALL LETTER S WITH ACUTE
015C	015D	LATIN CAPITAL LETTER S WITH CIRCUMFLEX	LATIN SMALL LETTER S WITH CIRCUMFLEX
015E	015F	LATIN CAPITAL LETTER S WITH CEDILLA	LATIN SMALL LETTER S WITH CEDILLA
0160	0161	LATIN CAPITAL LETTER S WITH CARON	LATIN SMALL LETTER S WITH CARON
0162	0163	LATIN CAPITAL LETTER T WITH CEDILLA	LATIN SMALL LETTER T WITH CEDILLA
0164	0165	LATIN CAPITAL LETTER T WITH CARON	LATIN SMALL LETTER T WITH CARON

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
0166	0167	LATIN CAPITAL LETTER T WITH STROKE	LATIN SMALL LETTER T WITH STROKE
0168	0169	LATIN CAPITAL LETTER U WITH TILDE	LATIN SMALL LETTER U WITH TILDE
016A	016B	LATIN CAPITAL LETTER U WITH MACRON	LATIN SMALL LETTER U WITH MACRON
016C	016D	LATIN CAPITAL LETTER U WITH BREVE	LATIN SMALL LETTER U WITH BREVE
016E	016F	LATIN CAPITAL LETTER U WITH RING ABOVE	LATIN SMALL LETTER U WITH RING ABOVE
0170	0171	LATIN CAPITAL LETTER U WITH DOUBLE ACUTE	LATIN SMALL LETTER U WITH DOUBLE ACUTE
0172	0173	LATIN CAPITAL LETTER U WITH OGONEK	LATIN SMALL LETTER U WITH OGONEK
0174	0175	LATIN CAPITAL LETTER W WITH CIRCUMFLEX	LATIN SMALL LETTER W WITH CIRCUMFLEX
0176	0177	LATIN CAPITAL LETTER Y WITH CIRCUMFLEX	LATIN SMALL LETTER Y WITH CIRCUMFLEX
0178	00FF	LATIN CAPITAL LETTER Y WITH DIAERESIS	LATIN SMALL LETTER Y DIAERESIS
0179	017A	LATIN CAPITAL LETTER Z WITH ACUTE	LATIN SMALL LETTER Z WITH ACUTE
017B	017C	LATIN CAPITAL LETTER Z WITH DOT ABOVE	LATIN SMALL LETTER Z WITH DOT ABOVE
017D	017E	LATIN CAPITAL LETTER Z WITH CARON	LATIN SMALL LETTER Z WITH CARON
0181	0253	LATIN CAPITAL LETTER B WITH HOOK	LATIN SMALL LETTER B WITH HOOK
0182	0183	LATIN CAPITAL LETTER B WITH TOPBAR	LATIN SMALL LETTER B WITH TOPBAR
0184	0185	LATIN CAPITAL LETTER TONE SIX	LATIN SMALL LETTER TONE SIX
0186	0254	LATIN CAPITAL LETTER OPEN O	LATIN SMALL LETTER OPEN O
0187	0188	LATIN CAPITAL LETTER C WITH HOOK	LATIN SMALL LETTER C WITH HOOK
018A	0257	LATIN CAPITAL LETTER D WITH HOOK	LATIN SMALL LETTER D WITH HOOK
018B	018C	LATIN CAPITAL LETTER D WITH TOPBAR	LATIN SMALL LETTER D WITH TOPBAR
018E	0258	LATIN CAPITAL LETTER REVERSED E	LATIN SMALL LETTER REVERSED E
018F	0259	LATIN CAPITAL LETTER SCHWA	LATIN SMALL LETTER SCHWA
0190	025B	LATIN CAPITAL LETTER OPEN E	LATIN SMALL LETTER OPEN E

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
0191	0192	LATIN CAPITAL LETTER F WITH HOOK	LATIN SMALL LETTER F WITH HOOK
0193	0260	LATIN CAPITAL LETTER G WITH HOOK	LATIN SMALL LETTER G WITH HOOK
0194	0263	LATIN CAPITAL LETTER GAMMA	LATIN SMALL LETTER GAMMA
0196	0269	LATIN CAPITAL LETTER IOTA	LATIN SMALL LETTER IOTA
0197	0268	LATIN CAPITAL LETTER I WITH STROKE	LATIN SMALL LETTER I WITH STROKE
0198	0199	LATIN CAPITAL LETTER K WITH HOOK	LATIN SMALL LETTER K WITH HOOK
019C	026f	LATIN CAPITAL LETTER TURNED M	LATIN SMALL LETTER TURNED M
019D	0272	LATIN CAPITAL LETTER N WITH LEFT HOOK	LATIN SMALL LETTER N WITH LEFT HOOK
019F	0275	LATIN CAPITAL LETTER O WITH MIDDLE TILDE	LATIN SMALL LETTER BARRED O
01A0	01A1	LATIN CAPITAL LETTER O WITH HORN	LATIN SMALL LETTER O WITH HORN
01A2	01A3	LATIN CAPITAL LETTER OI	LATIN SMALL LETTER OI
01A4	01A5	LATIN CAPITAL LETTER P WITH HOOK	LATIN SMALL LETTER P WITH HOOK
01A7	01A8	LATIN CAPITAL LETTER TONE TWO	LATIN SMALL LETTER TONE TWO
01A9	0283	LATIN CAPITAL LETTER ESH	LATIN SMALL LETTER ESH
01AC	01AD	LATIN CAPITAL LETTER T WITH HOOK	LATIN SMALL LETTER T WITH HOOK
01AE	0288	LATIN CAPITAL LETTER T WITH RETROFLEX HOOK	LATIN SMALL LETTER T WITH RETROFLEX HOOK
01AF	01B0	LATIN CAPITAL LETTER U WITH HORN	LATIN SMALL LETTER U WITH HORN
01B1	028A	LATIN CAPITAL LETTER UPSILON	LATIN SMALL LETTER UPSILON
01B2	028B	LATIN CAPITAL LETTER V WITH HOOK	LATIN SMALL LETTER V WITH HOOK
01B3	01B4	LATIN CAPITAL LETTER Y WITH HOOK	LATIN SMALL LETTER Y WITH HOOK
01B5	01B6	LATIN CAPITAL LETTER Z WITH STROKE	LATIN SMALL LETTER Z WITH STROKE
01B7	0292	LATIN CAPITAL LETTER EZH	LATIN SMALL LETTER EZH
01B8	01B9	LATIN CAPITAL LETTER EZH REVERSED	LATIN SMALL LETTER EZH REVERSED
01BC	01BD	LATIN CAPITAL LETTER TONE FIVE	LATIN SMALL LETTER TONE FIVE
01C4	01C6	LATIN CAPITAL LETTER DZ WITH CARON	LATIN SMALL LETTER DZ WITH CARON

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
01C5	01C6	LATIN CAPITAL LETTER D WITH SMALL LETTER Z WITH CARON	LATIN SMALL LETTER DZ WITH CARON
01C7	01C9	LATIN CAPITAL LETTER LJ	LATIN SMALL LETTER LJ
01C8	01C9	LATIN CAPITAL LETTER L WITH SMALL LETTER J	LATIN SMALL LETTER LJ
01CA	01CC	LATIN CAPITAL LETTER NJ	LATIN SMALL LETTER NJ
01CB	01CC	LATIN CAPITAL LETTER N WITH SMALL LETTER J	LATIN SMALL LETTER NJ
01CD	01CE	LATIN CAPITAL LETTER A WITH CARON	LATIN SMALL LETTER A WITH CARON
01CF	01D0	LATIN CAPITAL LETTER I WITH CARON	LATIN SMALL LETTER I WITH CARON
01D1	01D2	LATIN CAPITAL LETTER O WITH CARON	LATIN SMALL LETTER O WITH CARON
01D3	01D4	LATIN CAPITAL LETTER U WITH CARON	LATIN SMALL LETTER U WITH CARON
01D5	01D6	LATIN CAPITAL LETTER U WITH DIAERESIS AND MACRON	LATIN SMALL LETTER U WITH DIAERESIS AND MACRON
01D7	01D8	LATIN CAPITAL LETTER U WITH DIAERESIS AND ACUTE	LATIN SMALL LETTER U WITH DIAERESIS AND ACUTE
01D9	01DA	LATIN CAPITAL LETTER U WITH DIAERESIS AND CARON	LATIN SMALL LETTER U WITH DIAERESIS AND CARON
01DB	01DC	LATIN CAPITAL LETTER U WITH DIAERESIS AND GRAVE	LATIN SMALL LETTER U WITH DIAERESIS AND GRAVE
01DE	01DF	LATIN CAPITAL LETTER A WITH DIAERESIS AND MACRON	LATIN SMALL LETTER A WITH DIAERESIS AND MACRON
01E0	01E1	LATIN CAPITAL LETTER A WITH DOT ABOVE AND MACRON	LATIN SMALL LETTER A WITH DOT ABOVE AND MACRON
01E2	01E3	LATIN CAPITAL LIGATURE AE WITH MACRON	LATIN SMALL LIGATURE AE WITH MACRON
01E4	01E5	LATIN CAPITAL LETTER G WITH STROKE	LATIN SMALL LETTER G WITH STROKE
01E6	01E7	LATIN CAPITAL LETTER G WITH CARON	LATIN SMALL LETTER G WITH CARON
01E8	01E9	LATIN CAPITAL LETTER K WITH CARON	LATIN SMALL LETTER K WITH CARON
01EA	01EB	LATIN CAPITAL LETTER O WITH OGONEK	LATIN SMALL LETTER O WITH OGONEK
01EC	01ED	LATIN CAPITAL LETTER O WITH OGONEK AND MACRON	LATIN SMALL LETTER O WITH OGONEK AND MACRON
01EE	01EF	LATIN CAPITAL LETTER EZH WITH CARON	LATIN SMALL LETTER EZH WITH CARON
01F1	01F3	LATIN CAPITAL LETTER DZ	LATIN SMALL LETTER DZ
01F4	01F5	LATIN CAPITAL LETTER G WITH ACUTE	LATIN SMALL LETTER G WITH ACUTE

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
01FA	01FB	LATIN CAPITAL LETTER A WITH RING ABOVE AND ACUTE	LATIN SMALL LETTER A WITH RING ABOVE AND ACUTE
01FC	01FD	LATIN CAPITAL LIGATURE AE WITH ACUTE	LATIN SMALL LIGATURE AE WITH ACUTE
01FE	01FF	LATIN CAPITAL LETTER O WITH STROKE AND ACUTE	LATIN SMALL LETTER O WITH STROKE AND ACUTE
0200	0201	LATIN CAPITAL LETTER A WITH DOUBLE GRAVE	LATIN SMALL LETTER A WITH DOUBLE GRAVE
0202	0203	LATIN CAPITAL LETTER A WITH INVERTED BREVE	LATIN SMALL LETTER A WITH INVERTED BREVE
0204	0205	LATIN CAPITAL LETTER E WITH DOUBLE GRAVE	LATIN SMALL LETTER E WITH DOUBLE GRAVE
0206	0207	LATIN CAPITAL LETTER E WITH INVERTED BREVE	LATIN SMALL LETTER E WITH INVERTED BREVE
0208	0209	LATIN CAPITAL LETTER I WITH DOUBLE GRAVE	LATIN SMALL LETTER I WITH DOUBLE GRAVE
020A	020B	LATIN CAPITAL LETTER I WITH INVERTED BREVE	LATIN SMALL LETTER I WITH INVERTED BREVE
020C	020D	LATIN CAPITAL LETTER O WITH DOUBLE GRAVE	LATIN SMALL LETTER O WITH DOUBLE GRAVE
020E	020F	LATIN CAPITAL LETTER O WITH INVERTED BREVE	LATIN SMALL LETTER O WITH INVERTED BREVE
0210	0211	LATIN CAPITAL LETTER R WITH DOUBLE GRAVE	LATIN SMALL LETTER R WITH DOUBLE GRAVE
0212	0213	LATIN CAPITAL LETTER R WITH INVERTED BREVE	LATIN SMALL LETTER R WITH INVERTED BREVE
0214	0215	LATIN CAPITAL LETTER U WITH DOUBLE GRAVE	LATIN SMALL LETTER U WITH DOUBLE GRAVE
0216	0217	LATIN CAPITAL LETTER U WITH INVERTED BREVE	LATIN SMALL LETTER U WITH INVERTED BREVE
0386	03AC	GREEK CAPITAL LETTER ALPHA WITH TONOS	GREEK SMALL LETTER ALPHA WITH TONOS
0388	03AD	GREEK CAPITAL LETTER EPSILON WITH TONOS	GREEK SMALL LETTER EPSILON WITH TONOS
0389	03AE	GREEK CAPITAL LETTER ETA WITH TONOS	GREEK SMALL LETTER ETA WITH TONOS
038A	03AF	GREEK CAPITAL LETTER IOTA WITH TONOS	GREEK SMALL LETTER IOTA WITH TONOS
038C	03CC	GREEK CAPITAL LETTER OMICRON WITH TONOS	GREEK SMALL LETTER OMICRON WITH TONOS
038E	03CD	GREEK CAPITAL LETTER UPSILON WITH TONOS	GREEK SMALL LETTER UPSILON WITH TONOS
038F	03CE	GREEK CAPITAL LETTER OMEGA WITH TONOS	GREEK SMALL LETTER OMEGA WITH TONOS
0391	03B1	GREEK CAPITAL LETTER ALPHA	GREEK SMALL LETTER ALPHA
0392	03B2	GREEK CAPITAL LETTER BETA	GREEK SMALL LETTER BETA

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
0393	03B3	GREEK CAPITAL LETTER GAMMA	GREEK SMALL LETTER GAMMA
0394	03B4	GREEK CAPITAL LETTER DELTA	GREEK SMALL LETTER DELTA
0395	03B5	GREEK CAPITAL LETTER EPSILON	GREEK SMALL LETTER EPSILON
0396	03B6	GREEK CAPITAL LETTER ZETA	GREEK SMALL LETTER ZETA
0397	03B7	GREEK CAPITAL LETTER ETA	GREEK SMALL LETTER ETA
0398	03B8	GREEK CAPITAL LETTER THETA	GREEK SMALL LETTER THETA
0399	03B9	GREEK CAPITAL LETTER IOTA	GREEK SMALL LETTER IOTA
039A	03BA	GREEK CAPITAL LETTER KAPPA	GREEK SMALL LETTER KAPPA
039B	03BB	GREEK CAPITAL LETTER LAMDA	GREEK SMALL LETTER LAMDA
039C	03BC	GREEK CAPITAL LETTER MU	GREEK SMALL LETTER MU
039D	03BD	GREEK CAPITAL LETTER NU	GREEK SMALL LETTER NU
039E	03BE	GREEK CAPITAL LETTER XI	GREEK SMALL LETTER XI
039F	03BF	GREEK CAPITAL LETTER OMICRON	GREEK SMALL LETTER OMICRON
03A0	03C0	GREEK CAPITAL LETTER PI	GREEK SMALL LETTER PI
03A1	03C1	GREEK CAPITAL LETTER RHO	GREEK SMALL LETTER RHO
03A3	03C3	GREEK CAPITAL LETTER SIGMA	GREEK SMALL LETTER SIGMA
03A4	03C4	GREEK CAPITAL LETTER TAU	GREEK SMALL LETTER TAU
03A5	03C5	GREEK CAPITAL LETTER UPSILON	GREEK SMALL LETTER UPSILON
03A6	03C6	GREEK CAPITAL LETTER PHI	GREEK SMALL LETTER PHI
03A7	03C7	GREEK CAPITAL LETTER CHI	GREEK SMALL LETTER CHI
03A8	03C8	GREEK CAPITAL LETTER PSI	GREEK SMALL LETTER PSI
03A9	03C9	GREEK CAPITAL LETTER OMEGA	GREEK SMALL LETTER OMEGA
03AA	03CA	GREEK CAPITAL LETTER IOTA WITH DIALYTIKA	GREEK SMALL LETTER IOTA WITH DIALYTIKA
03AB	03CB	GREEK CAPITAL LETTER UPSILON WITH DIALYTIKA	GREEK SMALL LETTER UPSILON WITH DIALYTIKA
03E2	03E3	COPTIC CAPITAL LETTER SHEI	COPTIC SMALL LETTER SHEI
03E4	03E5	COPTIC CAPITAL LETTER FEI	COPTIC SMALL LETTER FEI
03E6	03E7	COPTIC CAPITAL LETTER KHEI	COPTIC SMALL LETTER KHEI
03E8	03E9	COPTIC CAPITAL LETTER HORI	COPTIC SMALL LETTER HORI
03EA	03EB	COPTIC CAPITAL LETTER GANGIA	COPTIC SMALL LETTER GANGIA
03EC	03ED	COPTIC CAPITAL LETTER SHIMA	COPTIC SMALL LETTER SHIMA

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
03EE	03EF	COPTIC CAPITAL LETTER DEI	COPTIC SMALL LETTER DEI
0401	0451	CYRILLIC CAPITAL LETTER IO	CYRILLIC SMALL LETTER IO
0402	0452	CYRILLIC CAPITAL LETTER DJE (SERBOCROATIAN)	CYRILLIC SMALL LETTER DJE (SERBOCROATIAN)
0403	0453	CYRILLIC CAPITAL LETTER GJE	CYRILLIC SMALL LETTER GJE
0404	0454	CYRILLIC CAPITAL LETTER UKRAINIAN IE	CYRILLIC SMALL LETTER UKRAINIAN IE
0405	0455	CYRILLIC CAPITAL LETTER DZE	CYRILLIC SMALL LETTER DZE
0406	0456	CYRILLIC CAPITAL LETTER BYELORUSSIAN_UKRAINIAN I	CYRILLIC SMALL LETTER BYELORUSSIAN-UKRAINIAN I
0407	0457	CYRILLIC CAPITAL LETTER YI (UKRANIAN)	CYRILLIC SMALL LETTER YI (UKRANIAN)
0408	0458	CYRILLIC CAPITAL LETTER JE	CYRILLIC SMALL LETTER JE
0409	0459	CYRILLIC CAPITAL LETTER LJE	CYRILLIC SMALL LETTER LJE
040A	045A	CYRILLIC CAPITAL LETTER NJE	CYRILLIC SMALL LETTER NJE
040B	045B	CYRILLIC CAPITAL LETTER TSHE (SERBOCROATIAN)	CYRILLIC SMALL LETTER TSHE (SERBOCROATIAN)
040C	045C	CYRILLIC CAPITAL LETTER KJE	CYRILLIC SMALL LETTER KJE
040E	045E	CYRILLIC CAPITAL LETTER SHORT U (BYELORUSSIAN)	CYRILLIC SMALL LETTER SHORT U (BYELORUSSIAN)
040F	045F	CYRILLIC CAPITAL LETTER DZHE	CYRILLIC SMALL LETTER DZHE
0410	0430	CYRILLIC CAPITAL LETTER A	CYRILLIC SMALL LETTER A
0411	0431	CYRILLIC CAPITAL LETTER BE	CYRILLIC SMALL LETTER BE
0412	0432	CYRILLIC CAPITAL LETTER VE	CYRILLIC SMALL LETTER VE
0413	0433	CYRILLIC CAPITAL LETTER GHE	CYRILLIC SMALL LETTER GHE
0414	0434	CYRILLIC CAPITAL LETTER DE	CYRILLIC SMALL LETTER DE
0415	0435	CYRILLIC CAPITAL LETTER IE	CYRILLIC SMALL LETTER IE
0416	0436	CYRILLIC CAPITAL LETTER ZHE	CYRILLIC SMALL LETTER ZHE
0417	0437	CYRILLIC CAPITAL LETTER ZE	CYRILLIC SMALL LETTER ZE
0418	0438	CYRILLIC CAPITAL LETTER I	CYRILLIC SMALL LETTER I
0419	0439	CYRILLIC CAPITAL LETTER SHORT I	CYRILLIC SMALL LETTER SHORT I
041A	043A	CYRILLIC CAPITAL LETTER KA	CYRILLIC SMALL LETTER KA
041B	043B	CYRILLIC CAPITAL LETTER EL	CYRILLIC SMALL LETTER EL
041C	043C	CYRILLIC CAPITAL LETTER EM	CYRILLIC SMALL LETTER EM
041D	043D	CYRILLIC CAPITAL LETTER EN	CYRILLIC SMALL LETTER EN
041E	043E	CYRILLIC CAPITAL LETTER O	CYRILLIC SMALL LETTER O
041F	043F	CYRILLIC CAPITAL LETTER PE	CYRILLIC SMALL LETTER PE

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
0420	0440	CYRILLIC CAPITAL LETTER ER	CYRILLIC SMALL LETTER ER
0421	0441	CYRILLIC CAPITAL LETTER ES	CYRILLIC SMALL LETTER ES
0422	0442	CYRILLIC CAPITAL LETTER TE	CYRILLIC SMALL LETTER TE
0423	0443	CYRILLIC CAPITAL LETTER U	CYRILLIC SMALL LETTER U
0424	0444	CYRILLIC CAPITAL LETTER EF	CYRILLIC SMALL LETTER EF
0425	0445	CYRILLIC CAPITAL LETTER HA	CYRILLIC SMALL LETTER HA
0426	0446	CYRILLIC CAPITAL LETTER TSE	CYRILLIC SMALL LETTER TSE
0427	0447	CYRILLIC CAPITAL LETTER CHE	CYRILLIC SMALL LETTER CHE
0428	0448	CYRILLIC CAPITAL LETTER SHA	CYRILLIC SMALL LETTER SHA
0429	0449	CYRILLIC CAPITAL LETTER SHCHA	CYRILLIC SMALL LETTER SHCHA
042A	044A	CYRILLIC CAPITAL LETTER HARD SIGN	CYRILLIC SMALL LETTER HARD SIGN
042B	044B	CYRILLIC CAPITAL LETTER YERU	CYRILLIC SMALL LETTER YERU
042C	044C	CYRILLIC CAPITAL LETTER SOFT SIGN	CYRILLIC SMALL LETTER SOFT SIGN
042D	044D	CYRILLIC CAPITAL LETTER E	CYRILLIC SMALL LETTER E
042E	044E	CYRILLIC CAPITAL LETTER YU	CYRILLIC SMALL LETTER YU
042F	044F	CYRILLIC CAPITAL LETTER YA	CYRILLIC SMALL LETTER YA
0460	0461	CYRILLIC CAPITAL LETTER OMEGA	CYRILLIC SMALL LETTER OMEGA
0462	0463	CYRILLIC CAPITAL LETTER YAT	CYRILLIC SMALL LETTER YAT
0464	0465	CYRILLIC CAPITAL LETTER IOTIFIED E	CYRILLIC SMALL LETTER IOTIFIED E
0466	0467	CYRILLIC CAPITAL LETTER LITTLE YUS	CYRILLIC SMALL LETTER LITTLE YUS
0468	0469	CYRILLIC CAPITAL LETTER IOTIFIED LITTLE YUS	CYRILLIC SMALL LETTER IOTFIED LITTLE YUS
046A	046B	CYRILLIC CAPITAL LETTER BIG YUS	CYRILLIC SMALL LETTER BIG YUS
046C	046D	CYRILLIC CAPITAL LETTER IOTIFIED BIG YUS	CYRILLIC SMALL LETTER IOTIFIED BIG YUS
046E	046F	CYRILLIC CAPITAL LETTER KSI	CYRILLIC SMALL LETTER KSI
0470	0471	CYRILLIC CAPITAL LETTER PSI	CYRILLIC SMALL LETTER PSI
0472	0473	CYRILLIC CAPITAL LETTER FITA	CYRILLIC SMALL LETTER FITA
0474	0475	CYRILLIC CAPITAL LETTER IZHITSA	CYRILLIC SMALL LETTER IZHITSA
0476	0477	CYRILLIC CAPITAL LETTER IZHITSA WITH DOUBLE GRAVE ACCENT	CYRILLIC SMALL LETTER IZHITSA WITH DOUBLE GRAVE ACCENT

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
0478	0479	CYRILLIC CAPITAL LETTER UK	CYRILLIC SMALL LETTER UK
047A	047B	CYRILLIC CAPITAL LETTER ROUND OMEGA	CYRILLIC SMALL LETTER ROUND OMEGA
047C	047D	CYRILLIC CAPITAL LETTER OMEGA WITH TITLO	CYRILLIC SMALL LETTER OMEGA WITH TITLO
047E	047F	CYRILLIC CAPITAL LETTER OT	CYRILLIC SMALL LETTER OT
0480	0481	CYRILLIC CAPITAL LETTER KOPPA	CYRILLIC SMALL LETTER KOPPA
0490	0491	CYRILLIC CAPITAL LETTER GHE WITH UPTURN	CYRILLIC SMALL LETTER GHE WITH UPTURN
0492	0493	CYRILLIC CAPITAL LETTER GHE WITH STROKE	CYRILLIC SMALL LETTER GHE WITH STROKE
0494	0495	CYRILLIC CAPITAL LETTER GHE WITH MIDDLE HOOK	CYRILLIC SMALL LETTER GHE WITH MIDDLE HOOK
0496	0497	CYRILLIC CAPITAL LETTER ZHE WITH DESCENDER	CYRILLIC SMALL LETTER ZHE WITH DESCENDER
0498	0499	CYRILLIC CAPITAL LETTER ZE WITH DESCENDER	CYRILLIC SMALL LETTER ZE WITH DESCENDER
049A	049B	CYRILLIC CAPITAL LETTER KA WITH DESCENDER	CYRILLIC SMALL LETTER KA WITH DESCENDER
049C	049D	CYRILLIC CAPITAL LETTER KA WITH VERTICAL STROKE	CYRILLIC SMALL LETTER KA WITH VERTICAL STROKE
049E	049F	CYRILLIC CAPITAL LETTER KA WITH STROKE	CYRILLIC SMALL LETTER KA WITH STROKE
04A0	04A1	CYRILLIC CAPITAL LETTER BASHKIR KA	CYRILLIC SMALL LETTER EASHKIR KA
04A2	04A3	CYRILLIC CAPITAL LETTER EN WITH DESCENDER	CYRILLIC SMALL LETTER EN WITH DESCENDER
04A4	04A5	CYRILLIC CAPITAL LIGATURE EN GHF	CYRILLIC SMALL LIGATURE EN GHE
04A6	04A7	CYRILLIC CAPITAL LETTER PE WITH MIDDLE HOOK (ABKHASIAN)	CYRILLIC SMALL LETTER PE WITH MIDDLE HOOK (ABKHASIAN)
04A8	04A9	CYRILLIC CAPITAL LETTER ABKHASIAN HA	CYRILLIC SMALL LETTER ABKHASIAN HA
04AA	04AB	CYRILLIC CAPITAL LETTER ES WITH DESCENDER	CYRILLIC SMALL LETTER ES WITH DESCENDER
04AC	04AD	CYRILLIC CAPITAL LETTER TE WITH DESCENDER	CYRILLIC SMALL LETTER TE WITH DESCENDER
04AE	04AF	CYRILLIC CAPITAL LETTER STRAIGHT U	CYRILLIC SMALL LETTER STRAIGHT U
04B0	04B1	CYRILLIC CAPITAL LETTER STRAIGHT U WITH STROKE	CYRILLIC SMALL LETTER STRAIGHT U WITH STROKE
04B2	04B3	CYRILLIC CAPITAL LETTER HA WITH DESCENDER	CYRILLIC SMALL LETTER HA WITH DESCENDER
04B4	04B5	CYRILLIC CAPITAL LIGATURE TE TSE (ABKHASIAN)	CYRILLIC SMALL LIGATURE TE TSE (ABKHASIAN)

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
04B6	04B7	CYRILLIC CAPITAL LETTER CHE WITH DESCENDER	CYRILLIC SMALL LETTER CHE WITH DESCENDER
04B8	04B9	CYRILLIC CAPITAL LETTER CHE WITH VERTICAL STROKE	CYRILLIC SMALL LETTER CHE WITH VERTICAL STROKE
04BA	04BB	CYRILLIC CAPITAL LETTER SHHA	CYRILLIC SMALL LETTER SHHA
04BC	04BD	CYRILLIC CAPITAL LETTER ABKHASIAN CHE	CYRILLIC SMALL LETTER ABKHASIAN CHE
04BE	04BF	CYRILLIC CAPITAL LETTER ABKHASIAN CHE WITH DESCENDER	CYRILLIC SMALL LETTER ABKHASIAN CHE WITH DESCENDER
04C1	04C2	CYRILLIC CAPITAL LETTER ZHE WITH BREVE	CYRILLIC SMALL LETTER ZHE WITH BREVE
04C3	04C4	CYRILLIC CAPITAL LETTER KA WITH HOOK	CYRILLIC SMALL LETTER KA WITH HOOK
04C7	04C8	CYRILLIC CAPITAL LETTER EN WITH HOOK	CYRILLIC SMALL LETTER EN WITH HOOK
04CB	04CC	CYRILLIC CAPITAL LETTER KHAKASSIAN CHE	CYRILLIC SMALL LETTER KHAKASSIAN CHE
04D0	04D1	CYRILLIC CAPITAL LETTER A WITH BREVE	CYRILLIC SMALL LETTER A WITH BREVE
04D2	04D3	CYRILLIC CAPITAL LETTER A WITH DIAERESIS	CYRILLIC SMALL LETTER A WITH DIAERESIS
04D4	04D5	CYRILLIC CAPITAL LIGATURE A IE	CYRILLIC SMALL LIGATURE A IE
04D6	04D7	CYRILLIC CAPITAL LETTER IE WITH BREVE	CYRILLIC SMALL LETTER IE WITH BREVE
04D8	04D9	CYRILLIC CAPITAL LETTER SCHWA	CYRILLIC SMALL LETTER SCHWA
04DA	04DB	CYRILLIC CAPITAL LETTER SCHWA WITH DIAERESIS	CYRILLIC SMALL LETTER SCHWA WITH DIAERESIS
04DC	04DD	CYRILLIC CAPITAL LETTER ZHE WITH DIAERESIS	CYRILLIC SMALL LETTER ZHE WITH DIAERESIS
04DE	04DF	CYRILLIC CAPITAL LETTER ZE WITH DIAERESIS	CYRILLIC SMALL LETTER ZE WITH DIAERESIS
04E0	04E1	CYRILLIC CAPITAL LETTER ABKHASIAN DZE	CYRILLIC SMALL LETTER ABKHASIAN DZE
04E2	04E3	CYRILLIC CAPITAL LETTER I WITH MACRON	CYRILLIC SMALL LETTER I WITH MACRON
04E4	04E5	CYRILLIC CAPITAL LETTER I WITH DIAERESIS	CYRILLIC SMALL LETTER I WITH DIAERESIS
04E6	04E7	CYRILLIC CAPITAL LETTER O WITH DIAERESIS	CYRILLIC SMALL LETTER O WITH DIAERESIS
04E8	04E9	CYRILLIC CAPITAL LETTER BARRED O	CYRILLIC SMALL LETTER BARRED O
04EA	04EB	CYRILLIC CAPITAL LETTER BARRED O WITH DIAERESIS	CYRILLIC SMALL LETTER BARRED 0 WITH DIAERESIS

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
04EE	04EF	CYRILLIC CAPITAL LETTER U WITH MACRON	CYRILLIC SMALL LETTER U WITH MACRON
04F0	04F1	CYRILLIC CAPITAL LETTER U WITH DIAERESIS	CYRILLIC SMALL LETTER U WITH DIAERESIS
04F2	04F3	CYRILLIC CAPITAL LETTER U WITH DOUBLE ACUTE	CYRILLIC SMALL LETTER U WITH DOUBLE ACUTE
04F4	04F5	CYRILLIC CAPITAL LETTER CHE WITH DIAERESIS	CYRILLIC SMALL LETTER CHE WITH DIAERESIS
04F8	04F9	CYRILLIC CAPITAL LETTER YERU WITH DIAERESIS	CYRILLIC SMALL LETTER YERU WITH DIAERESIS
0531	0561	ARMENIAN CAPITAL LETTER AYB	ARMENIAN SMALL LETTER AYB
0532	0562	ARMENIAN CAPITAL LETTER BEN	ARMENIAN SMALL LETTER BEN
0533	0563	ARMENIAN CAPITAL LETTER GIM	ARMENIAN SMALL LETTER GIM
0534	0564	ARMENIAN CAPITAL LETTER DA	ARMENIAN SMALL LETTER DA
0535	0565	ARMENIAN CAPITAL LETTER ECH	ARMENIAN SMALL LETTER ECH
0536	0566	ARMENIAN CAPITAL LETTER ZA	ARMENIAN SMALL LETTER ZA
0537	0567	ARMENIAN CAPITAL LETTER EH	ARMENIAN SMALL LETTER EH
0538	0568	ARMENIAN CAPITAL LETTER ET	ARMENIAN SMALL LETTER ET
0539	0569	ARMENIAN CAPITAL LETTER TO	ARMENIAN SMALL LETTER TO
053A	056A	ARMENIAN CAPITAL LETTER ZHE	ARMENIAN SMALL LETTER ZHE
053B	056B	ARMENIAN CAPITAL LETTER INI	ARMENIAN SMALL LETTER INI
053C	056C	ARMENIAN CAPITAL LETTER LIWN	ARMENIAN SMALL LETTER LIWN
053D	056D	ARMENIAN CAPITAL LETTER XEH	ARMENIAN SMALL LETTER XEH
053E	056E	ARMENIAN CAPITAL LETTER CA	ARMENIAN SMALL LETTER CA
053F	056F	ARMENIAN CAPITAL LETTER KEN	ARMENIAN SMALL LETTER KEN
0540	0570	ARMENIAN CAPITAL LETTER HO	ARMENIAN SMALL LETTER HO
0541	0571	ARMENIAN CAPITAL LETTER JA	ARMENIAN SMALL LETTER JA
0542	0572	ARMENIAN CAPITAL LETTER GHAD	ARMENIAN SMALL LETTER GHAD

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
0543	0573	ARMENIAN CAPITAL LETTER CHEH	ARMENIAN SMALL LETTER CHEH
0544	0574	ARMENIAN CAPITAL LETTER MEN	ARMENIAN SMALL LETTER MEN
0545	0575	ARMENIAN CAPITAL LETTER YI	ARMENIAN SMALL LETTER YI
0546	0576	ARMENIAN CAPITAL LETTER NOW	ARMENIAN SMALL LETTER NOW
0547	0577	ARMENIAN CAPITAL LETTER SHA	ARMENIAN SMALL LETTER SNA
0548	0578	ARMENIAN CAPITAL LETTER VO	ARMENIAN SMALL LETTER VO
0549	0579	ARMENIAN CAPITAL LETTER CHA	ARMENIAN SMALL LETTER CHA
054A	057A	ARMENIAN CAPITAL LETTER PEH	ARMENIAN SMALL LETTER PEH
054B	057B	ARMENIAN CAPITAL LETTER JHEH	ARMENIAN SMALL LETTER JHEH
054C	057C	ARMENIAN CAPITAL LETTER RA	ARMENIAN SMALL LETTER RA
054D	057D	ARMENIAN CAPITAL LETTER SEH	ARMENIAN SMALL LETTER SEH
054E	057E	ARMENIAN CAPITAL LETTER VEW	ARMENIAN SMALL LETTER VEW
054F	057F	ARMENIAN CAPITAL LETTER TIWN	ARMENIAN SMALL LETTER TIWN
0550	0580	ARMENIAN CAPITAL LETTER REH	ARMENIAN SMALL LETTER REH
0551	0581	ARMENIAN CAPITAL LETTER CO	ARMENIAN SMALL LETTER CO
0552	0582	ARMENIAN CAPITAL LETTER YIWN	ARMENIAN SMALL LETTER YIWN
0553	0583	ARMENIAN CAPITAL LETTER PIWR	ARMENIAN SMALL LETTER PIWP
0554	0584	ARMENIAN CAPITAL LETTER KEH	ARMENIAN SMALL LETTER KEH
0555	0585	ARMENIAN CAPITAL LETTER OH	ARMENIAN SMALL LETTER OH
0556	0586	ARMENIAN CAPITAL LETTER FEH	ARMENIAN SMALL LETTER FEH
10A0	10D0	GEORGIAN CAPITAL LETTER AN (KHUTSURI)	GEORGIAN LETTER AN
10A1	10D1	GEORGIAN CAPITAL LETTER BAN (KHUTSURI)	GEORGIAN LETTER BAN
10A2	10D2	GEORGIAN CAPITAL LETTER GAN (KHUTSURI)	GEORGIAN LETTER GAN
10A3	10D3	GEORGIAN CAPITAL LETTER DON (KHUTSURI)	GEORGIAN LETTER DON

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
10A4	10D4	GEORGIAN CAPITAL LETTER EN (KHUTSURI)	GEORGIAN LETTER EN
10A5	10D5	GEORGIAN CAPITAL LETTER VIN (KHUTSURI)	GEORGIAN LETTER VIN
10A6	10D6	GEORGIAN CAPITAL LETTER ZEN (KHUTSURI)	GEORGIAN LETTER ZEN
10A7	10D7	GEORGIAN CAPITAL LETTER TAN (KHUTSURI)	GEORGIAN LETTER TAN
10A8	10D8	GEORGIAN CAPITAL LETTER IN (KHUTSURI)	GEORGIAN LETTER IN
10A9	10D9	GEORGIAN CAPITAL LETTER KAN (KHUTSURI)	GEORGIAN LETTER KAN
10AA	10DA	GEORGIAN CAPITAL LETTER LAS (KHUTSURI)	GEORGIAN LETTER LAS
10AB	10DB	GEORGIAN CAPITAL LETTER MAN (KHUTSURI)	GEORGIAN LETTER MAN
10AC	10DC	GEORGIAN CAPITAL LETTER NAR (KHUTSURI)	GEORGIAN LETTER NAR
10AD	10DD	GEORGIAN CAPITAL LETTER ON (KHUTSURI)	GEORGIAN LETTER ON
10AE	10DE	GEORGIAN CAPITAL LETTER PAR (KHUTSURI)	GEORGIAN LETTER PAR
10AF	10DF	GEORGIAN CAPITAL LETTER ZHAR (KHUTSURI)	GEORGIAN LETTER ZHAR
10B0	10E0	GEORGIAN CAPITAL LETTER RAE (KHUTSURI)	GEORGIAN LETTER RAE
10B1	10E1	GEORGIAN CAPITAL LETTER SAN (KHUTSURI)	GEORGIAN LETTER SAN
10B2	10E2	GEORGIAN CAPITAL LETTER TAR (KHUTSURI)	GEORGIAN LETTER TAR
10B3	10E3	GEORGIAN CAPITAL LETTER UN (KHUTSURI)	GEORGIAN LETTER UN
10B4	10E4	GEORGIAN CAPITAL LETTER PHAR (KHUTSURI)	GEORGIAN LETTER PHAR
10B5	10E5	GEORGIAN CAPITAL LETTER KHAR (KHUTSURI)	GEORGIAN LETTER KHAR
10B6	10E6	GEORGIAN CAPITAL LETTER GHAN (KHUTSURI)	GEORGIAN LETTER GHAN
10B7	10E7	GEORGIAN CAPITAL LETTER QAR (KHUTSURI)	GEORGIAN LETTER QAR
10B8	10E8	GEORGIAN CAPITAL LETTER SHIN (KHUTSURI)	GEORGIAN LETTER SHIN
10B9	10E9	GEORGIAN CAPITAL LETTER CHIN (KHUTSURI)	GEORGIAN LETTER CHIN
10BA	10EA	GEORGIAN CAPITAL LETTER CAN (KHUTSURI)	GEORGIAN LETTER CAN

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
10BB	10EB	GEORGIAN CAPITAL LETTER JIL (KHUTSURI)	GEORGIAN LETTER JIL
10BC	10EC	GEORGIAN CAPITAL LETTER CIL (KHUTSURI)	GEORGIAN LETTER CIL
10BD	10ED	GEORGIAN CAPITAL LETTER CHAR (KHUTSURI)	GEORGIAN LETTER CHAR
10BE	10EE	GEORGIAN CAPITAL LETTER XAN (KHUTSURI)	GEORGIAN LETTER XAN
10BF	10EF	GEORGIAN CAPITAL LETTER JHAN (KHUTSURI)	GEORGIAN LETTER JHAN
10C0	10F0	GEORGIAN CAPITAL LETTER HAE (KHUTSURI)	GEORGIAN LETTER HAE
10C1	10F1	GEORGIAN CAPITAL LETTER HE (KHUTSURI)	GEORGIAN LETTER HE
10C2	10F2	GEORGIAN CAPITAL LETTER HIE (KHUTSURI)	GEORGIAN LETTER HIE
10C3	10F3	GEORGIAN CAPITAL LETTER WE (KHUTSURI)	GEORGIAN LETTER WE
10C4	10F4	GEORGIAN CAPITAL LETTER HAR (KHUTSURI)	GEORGIAN LETTER HAR
10C5	10F5	GEORGIAN CAPITAL LETTER HOE (KHUTSURI)	GEORGIAN LETTER HOE
1E00	1E01	LATIN CAPITAL LETTER A WITH RING BELOW	LATIN SMALL LETTER A WITH RING BELOW
1E02	1E03	LATIN CAPITAL LETTER B WITH DOT ABOVE	LATIN SMALL LETTER B WITH DOT ABOVE
1E04	1E05	LATIN CAPITAL LETTER B WITH DOT BELOW	LATIN SMALL LETTER B WITH DOT BELOW
1E06	1E07	LATIN CAPITAL LETTER B WITH LINE BELOW	LATIN SMALL LETTER B WITH LINE BELOW
1E08	1E09	LATIN CAPITAL LETTER C WITH CEDILLA AND ACUTE	LATIN SMALL LETTER C WITH CEDILLA AND ACUTE
1E0A	1E0B	LATIN CAPITAL LETTER D WITH DOT ABOVE	LATIN SMALL LETTER D WITH DOT ABOVE
1E0C	1E0D	LATIN CAPITAL LETTER D WITH DOT BELOW	LATIN SMALL LETTER D WITH DOT BELOW
1E0E	1E0F	LATIN CAPITAL LETTER D WITH LINE BELOW	LATIN SMALL LETTER D WITH LINE BELOW
1E10	1E11	LATIN CAPITAL LETTER D WITH CEDILLA	LATIN SMALL LETTER D WITH CEDILLA
1E12	1E13	LATIN CAPITAL LETTER D WITH CIRCUMFLEX BELOW	LATIN SMALL LETTER D WITH CIRCUMFLEX BELOW
1E14	1E15	LATIN CAPITAL LETTER E WITH MACRON AND GRAVE	LATIN SMALL LETTER E WITH MACRON AND GRAVE
1E16	1E17	LATIN CAPITAL LETTER E WITH MACRON AND ACUTE	LATIN SMALL LETTER E WITH MACRON AND ACUTE

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
1E18	1E19	LATIN CAPITAL LETTER E WITH CIRCUMFLEX BELOW	LATIN SMALL LETTER E WITH CIRCUMFLEX BELOW
1E1A	1E1B	LATIN CAPITAL LETTER E WITH TILDE BELOW	LATIN SMALL LETTER E WITH TILDE BELOW
1E1C	1E1D	LATIN CAPITAL LETTER E WITH CEDILLA AND BREVE	LATIN SMALL LETTER E WITH CEDILLA AND BREVE
1E1E	1E1F	LATIN CAPITAL LETTER F WITH DOT ABOVE	LATIN SMALL LETTER F WITH DOT ABOVE
1E20	1E21	LATIN CAPITAL LETTER G WITH MACRON	LATIN SMALL LETTER G WITH MACRON
1E22	1E23	LATIN CAPITAL LETTER H WITH DOT ABOVE	LATIN SMALL LETTER H WITH DOT ABOVE
1E24	1E25	LATIN CAPITAL LETTER H WITH DOT BELOW	LATIN SMALL LETTER H WITH DOT BELOW
1E26	1E27	LATIN CAPITAL LETTER H WITH DIAERESIS	LATIN SMALL LETTER H WITH DIAERESIS
1E28	1E29	LATIN CAPITAL LETTER H WITH CEDILLA	LATIN SMALL LETTER H WITH CEDILLA
1E2A	1E2B	LATIN CAPITAL LETTER H WITH BREVE BELOW	LATIN SMALL LETTER H WITH BREVE BELOW
1E2C	1E2D	LATIN CAPITAL LETTER I WITH TILDE BELOW	LATIN SMALL LETTER I WITH TILDE BELOW
1E2E	1E2F	LATIN CAPITAL LETTER I WITH DIAERESIS AND ACUTE	LATIN SMALL LETTER I WITH DIAERESIS AND ACUTE
1E30	1E31	LATIN CAPITAL LETTER K WITH ACUTE	LATIN SMALL LETTER K WITH ACUTE
1E32	1E33	LATIN CAPITAL LETTER K WITH DOT BELOW	LATIN SMALL LETTER K WITH DOT BELOW
1E34	1E35	LATIN CAPITAL LETTER K WITH LINE BELOW	LATIN SMALL LETTER K WITH LINE BELOW
1E36	1E37	LATIN CAPITAL LETTER L WITH DOT BELOW	LATIN SMALL LETTER L WITH DOT BELOW
1E38	1E39	LATIN CAPITAL LETTER L WITH DOT BELOW AND MACRON	LATIN SMALL LETTER L WITH DOT BELOW AND MACRON
1E3A	1E3B	LATIN CAPITAL LETTER L WITH LINE BELOW	LATIN SMALL LETTER L WITH LINE BELOW
1E3C	1E3D	LATIN CAPITAL LETTER L WITH CIRCUMFLEX BELOW	LATIN SMALL LETTER L WITH CIRCUMFLEX BELOW
1E3E	1E3F	LATIN CAPITAL LETTER M WITH ACUTE	LATIN SMALL LETTER M WITH ACUTE
1E40	1E41	LATIN CAPITAL LETTER M WITH DOT ABOVE	LATIN SMALL LETTER M WITH DOT ABOVE
1E42	1E43	LATIN CAPITAL LETTER M WITH DOT BELOW	LATIN SMALL LETTER M WITH DOT BELOW
1E44	1E45	LATIN CAPITAL LETTER N WITH DOT ABOVE	LATIN SMALL LETTER N WITH DOT ABOVE

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
1E46	1E47	LATIN CAPITAL LETTER N WITH DOT BELOW	LATIN SMALL LETTER N WITH DOT BELOW
1E48	1E49	LATIN CAPITAL LETTER N WITH LINE BELOW	LATIN SMALL LETTER N WITH LINE BELOW
1E4A	1E4B	LATIN CAPITAL LETTER N WITH CIRCUMFLEX BELOW	LATIN SMALL LETTER N WITH CIRCUMFLEX BELOW
1E4C	1E4D	LATIN CAPITAL LETTER O WITH TILDE AND ACUTE	LATIN SMALL LETTER O WITH TILDE AND ACUTE
1E4E	1E4F	LATIN CAPITAL LETTER O WITH TILDE AND DIAERESIS	LATIN SMALL LETTER O WITH TILDE AND DIAERESIS
1E50	1E51	LATIN CAPITAL LETTER O WITH MACRON AND GRAVE	LATIN SMALL LETTER O WITH MACRON AND GRAVE
1E52	1E53	LATIN CAPITAL LETTER O WITH MACRON AND ACUTE	LATIN SMALL LETTER O WITH MACRON AND ACUTE
1E54	1E55	LATIN CAPITAL LETTER P WITH ACUTE	LATIN SMALL LETTER P WITH ACUTE
1E56	1E57	LATIN CAPITAL LETTER P WITH DOT ABOVE	LATIN SMALL LETTER P WITH DOT ABOVE
1E58	1E59	LATIN CAPITAL LETTER R WITH DOT ABOVE	LATIN SMALL LETTER R WITH DOT ABOVE
1E5A	1E5B	LATIN CAPITAL LETTER R WITH DOT BELOW	LATIN SMALL LETTER R WITH DOT BELOW
1E5C	1E5D	LATIN CAPITAL LETTER R WITH DOT BELOW AND MACRON	LATIN SMALL LETTER R WITH DOT BELOW AND MACRON
1E5E	1E5F	LATIN CAPITAL LETTER R WITH LINE BELOW	LATIN SMALL LETTER R WITH LINE BELOW
1E60	1E61	LATIN CAPITAL LETTER S WITH DOT ABOVE	LATIN SMALL LETTER S WITH DOT ABOVE
1E62	1E63	LATIN CAPITAL LETTER S WITH DOT BELOW	LATIN SMALL LETTER S WITH DOT BELOW
1E64	1E65	LATIN CAPITAL LETTER S WITH ACUTE AND DOT ABOVE	LATIN SMALL LETTER S WITH ACUTE AND DOT ABOVE
1E66	1E67	LATIN CAPITAL LETTER S WITH CARON AND DOT ABOVE	LATIN SMALL LETTER S WITH CARON AND DOT ABOVE
1E68	1E69	LATIN CAPITAL LETTER S WITH DOT BELOW AND DOT ABOVE	LATIN SMALL LETTER S WITH DOT BELOW AND DOT ABOVE
1E6A	1E6B	LATIN CAPITAL LETTER T WITH DOT ABOVE	LATIN SMALL LETTER T WITH DOT ABOVE
1E6C	1E6D	LATIN CAPITAL LETTER T WITH DOT BELOW	LATIN SMALL LETTER T WITH DOT BELOW
1E6E	1E6F	LATIN CAPITAL LETTER T WITH LINE BELOW	LATIN SMALL LETTER T WITH LINE BELOW
1E70	1E71	LATIN CAPITAL LETTER T WITH CIRCUMFLEX BELOW	LATIN SMALL LETTER T WITH CIRCUMFLEX BELOW
1E72	1E73	LATIN CAPITAL LETTER U WITH DIAERESIS BELOW	LATIN SMALL LETTER U WITH DIAERESIS BELOW

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
1E74	1E75	LATIN CAPITAL LETTER U WITH TILDE BELOW	LATIN SMALL LETTER U WITH TILDE BELOW
1E76	1E77	LATIN CAPITAL LETTER U WITH CIRCUMFLEX BELOW	LATIN SMALL LETTER U WITH CIRCUMFLEX BELOW
1E78	1E79	LATIN CAPITAL LETTER U WITH TILDE AND ACUTE	LATIN SMALL LETTER U WITH TILDE AND ACUTE
1E7A	1E7B	LATIN CAPITAL LETTER U WITH MACRON AND DIAERESIS	LATIN SMALL LETTER U WITH MACRON AND DIAERESIS
1E7C	1E7D	LATIN CAPITAL LETTER V WITH TILDE	LATIN SMALL LETTER V WITH TILDE
1E7E	1E7F	LATIN CAPITAL LETTER V WITH DOT BELOW	LATIN SMALL LETTER V WITH DOT BELOW
1E80	1E81	LATIN CAPITAL LETTER W WITH GRAVE	LATIN SMALL LETTER W WITH GRAVE
1E82	1E83	LATIN CAPITAL LETTER W WITH ACUTE	LATIN SMALL LETTER W WITH ACUTE
1E84	1E85	LATIN CAPITAL LETTER W WITH DIAERESIS	LATIN SMALL LETTER W WITH DIAERESIS
1E86	1E87	LATIN CAPITAL LETTER W WITH DOT ABOVE	LATIN SMALL LETTER W WITH DOT ABOVE
1E88	1E89	LATIN CAPITAL LETTER W WITH DOT BELOW	LATIN SMALL LETTER W WITH DOT BELOW
1E8A	1E8B	LATIN CAPITAL LETTER X WITH DOT ABOVE	LATIN SMALL LETTER X WITH DOT ABOVE
1E8C	1E8D	LATIN CAPITAL LETTER X5 WITH DIAERESIS	LATIN SMALL LETTER X WITH DIAERESIS
1E8E	1E8F	LATIN CAPITAL LETTER Y WITH DOT ABOVE	LATIN SMALL LETTER Y WITH DOT ABOVE
1E90	1E91	LATIN CAPITAL LETTER Z WITH CIRCUMFLEX	LATIN SMALL LETTER Z WITH CIRCUMFLEX
1E92	1E93	LATIN CAPITAL LETTER Z WITH DOT BELOW	LATIN SMALL LETTER Z WITH DOT BELOW
1E94	1E95	LATIN CAPITAL LETTER Z WITH LINE BELOW	LATIN SMALL LETTER Z WITH LINE BELOW
1EA0	1EA1	LATIN CAPITAL LETTER A WITH DOT BELOW	LATIN SMALL LETTER A WITH DOT BELOW
1EA2	1EA3	LATIN CAPITAL LETTER A WITH HOOK ABOVE	LATIN SMALL LETTER A WITH HOOK ABOVE
1EA4	1EA5	LATIN CAPITAL LETTER A WITH CIRCUMFLEX AND ACUTE	LATIN SMALL LETTER A WITH CIRCUMFLEX AND ACUTE
1EA6	1EA7	LATIN CAPITAL LETTER A WITH CIRCUMFLEX AND GRAVE	LATIN SMALL LETTER A WITH CIRCUMFLEX AND GRAVE
1EA8	1EA9	LATIN CAPITAL LETTER A WITH CIRCUMFLEX AND HOOK ABOVE	LATIN SMALL LETTER A WITH CIRCUMFLEX AND HOOK ABOVE
1EAA	1EAB	LATIN CAPITAL LETTER A WITH CIRCUMFLEX AND TILDE	LATIN SMALL LETTER A WITH CIRCUMFLEX AND TILDE

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
1EAC	1EAD	LATIN CAPITAL LETTER A WITH CIRCUMFLEX AND DOT BELOW	LATIN SMALL LETTER A WITH CIRCUMFLEX AND DOT BELOW
1EAE	1EAF	LATIN CAPITAL LETTER A WITH BREVE AND ACUTE	LATIN SMALL LETTER A WITH BREVE AND ACUTE
1EB0	1EB1	LATIN CAPITAL LETTER A WITH BREVE AND GRAVE	LATIN SMALL LETTER A WITH BREVE AND GRAVE
1EB2	1EB3	LATIN CAPITAL LETTER A WITH BREVE AND HOOK ABOVE	LATIN SMALL LETTER A WITH BREVE AND HOOK ABOVE
1EB4	1EB5	LATIN CAPITAL LETTER A WITH BREVE AND TILDE	LATIN SMALL LETTER A WITH BREVE AND TILDE
1EB6	1EB7	LATIN CAPITAL LETTER A WITH BREVE AND DOT BELOW	LATIN SMALL LETTER A WITH BREVE AND DOT BELOW
1EB8	1EB9	LATIN CAPITAL LETTER E WITH DOT BELOW	LATIN SMALL LETTER E WITH DOT BELOW
1EBA	1EBB	LATIN CAPITAL LETTER E WITH HOOK ABOVE	LATIN SMALL LETTER E WITH HOOK ABOVE
1EBC	1EBD	LATIN CAPITAL LETTER E WITH TILDE	LATIN SMALL LETTER E WITH TILDE
1EBE	1EBF	LATIN CAPITAL LETTER E WITH CIRCUMFLEX AND ACUTE	LATIN SMALL LETTER E WITH CIRCUMFLEX AND ACUTE
1EC0	1EC1	LATIN CAPITAL LETTER E WITH CIRCUMFLEX AND GRAVE	LATIN SMALL LETTER E WITH CIRCUMFLEX AND GRAVE
1EC2	1EC3	LATIN CAPITAL LETTER E WITH CIRCUMFLEX AND HOOK ABOVE	LATIN SMALL LETTER E WITH CIRCUMFLEX AND HOOK ABOVE
1EC4	1EC5	LATIN CAPITAL LETTER E WITH CIRCUMFLEX AND TILDE	LATIN SMALL LETTER E WITH CIRCUMFLEX AND TILDE
1EC6	1EC7	LATIN CAPITAL LETTER E WITH CIRCUMFLEX AND DOT BELOW	LATIN SMALL LETTER E WITH CIRCUMFLEX AND DOT BELOW
1EC8	1EC9	LATIN CAPITAL LETTER I WITH HOOK ABOVE	LATIN SMALL LETTER I WITH HOOK ABOVE
1ECA	1ECB	LATIN CAPITAL LETTER I WITH DOT BELOW	LATIN SMALL LETTER I WITH DOT BELOW
1ECC	1ECD	LATIN CAPITAL LETTER O WITH DOT BELOW	LATIN SMALL LETTER O WITH DOT BELOW
1ECE	1ECF	LATIN CAPITAL LETTER O WITH HOOK ABOVE	LATIN SMALL LETTER O WITH HOOK ABOVE
1ED0	1ED1	LATIN CAPITAL LETTER O WITH CIRCUMFLEX AND ACUTE	LATIN SMALL LETTER O WITH CIRCUMFLEX AND ACUTE
1ED2	1ED3	LATIN CAPITAL LETTER O WITH CIRCUMFLEX AND GRAVE	LATIN SMALL LETTER O WITH CIRCUMFLEX AND GRAVE
1ED4	1ED5	LATIN CAPITAL LETTER O WITH CIRCUMFLEX AND HOOK ABOVE	LATIN SMALL LETTER O WITH CIRCUMFLEX AND HOOK ABOVE

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
1ED6	1ED7	LATIN CAPITAL LETTER O WITH CIRCUMFLEX AND TILDE	LATIN SMALL LETTER O WITH CIRCUMFLEX AND TILDE
1ED8	1ED9	LATIN CAPITAL LETTER O WITH CIRCUMFLEX AND DOT BELOW	LATIN SMALL LETTER O WITH CIRCUMFLEX AND DOT BELOW
1EDA	1EDB	LATIN CAPITAL LETTER O WITH HORN AND ACUTE	LATIN SMALL LETTER O WITH HORN AND ACUTE
1EDC	1EDD	LATIN CAPITAL LETTER O WITH HORN AND GRAVE	LATIN SMALL LETTER O WITH HORN AND GRAVE
1EDE	1EDF	LATIN CAPITAL LETTER O WITH HORN AND HOOK ABOVE	LATIN SMALL LETTER O WITH HORN AND HOOK ABOVE
1EE0	1EE1	LATIN CAPITAL LETTER O WITH HORN AND TILDE	LATIN SMALL LETTER O WITH HORN AND TILDE
1EE2	1EE3	LATIN CAPITAL LETTER O WITH HORN AND DOT BELOW	LATIN SMALL LETTER O WITH HORN AND DOT BELOW
1EE4	1EE5	LATIN CAPITAL LETTER U WITH DOT BELOW	LATIN SMALL LETTER U WITH DOT BELOW
1EE6	1EE7	LATIN CAPITAL LETTER U WITH HOOK ABOVE	LATIN SMALL LETTER U WITH HOOK ABOVE
1EE8	1EE9	LATIN CAPITAL LETTER U WITH HORN AND ACUTE	LATIN SMALL LETTER U WITH HORN AND ACUTE
1EEA	1EEB	LATIN CAPITAL LETTER U WITH HORN AND GRAVE	LATIN SMALL LETTER U WITH HORN AND GRAVE
1EEC	1EED	LATIN CAPITAL LETTER U WITH HORN AND HOOK ABOVE	LATIN SMALL LETTER U WITH HORN AND HOOK ABOVE
1EEE	1EEF	LATIN CAPITAL LETTER U WITH HORN AND TILDE	LATIN SMALL LETTER U WITH HORN AND TILDE
1EF0	1EF1	LATIN CAPITAL LETTER U WITH HORN AND DOT BELOW	LATIN SMALL LETTER U WITH HORN AND DOT BELOW
1EF2	1EF3	LATIN CAPITAL LETTER Y WITH GRAVE	LATIN SMALL LETTER Y WITH GRAVE
1EF4	1EF5	LATIN CAPITAL LETTER Y WITH DOT BELOW	LATIN SMALL LETTER Y WITH DOT BELOW
1EF6	1EF7	LATIN CAPITAL LETTER Y WITH HOOK ABOVE	LATIN SMALL LETTER Y WITH HOOK ABOVE
1EF8	1EF9	LATIN CAPITAL LETTER Y WITH TILDE	LATIN SMALL LETTER Y WITH TILDE
1F08	1F00	GREEK CAPITAL LETTER ALPHA WITH PSILI	GREEK SMALL LETTER ALPHA WITH PSILI
1F09	1F01	GREEK CAPITAL LETTER ALPHA WITH DASIA	GREEK SMALL LETTER ALPHA WITH DASIA
1F0A	1F02	GREEK CAPITAL LETTER ALPHA WITH PSILI AND VARIA	GREEK SMALL LETTER ALPHA WITH PSILI AND VARIA
1F0B	1F03	GREEK CAPITAL LETTER ALPHA WITH DASIA AND VARIA	GREEK SMALL LETTER ALPHA WITH DASIA AND VARIA
1F0C	1F04	GREEK CAPITAL LETTER ALPHA WITH PSILI AND OXIA	GREEK SMALL LETTER ALPHA WITH PSILI AND OXIA

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
1F0D	1F05	GREEK CAPITAL LETTER ALPHA WITH DASIA AND OXIA	GREEK SMALL LETTER ALPHA WITH DASIA AND OXIA
1F0E	1F06	GREEK CAPITAL LETTER ALPHA WITH PSILI AND PERISPOMENI	GREEK SMALL LETTER ALPHA WITH PSILI AND PERISPOMENI
1F0F	1F07	GREEK CAPITAL LETTER ALPHA WITH DASIA AND PERISPOMENI	GREEK SMALL LETTER ALPHA WITH DASIA AND PERISPOMENI
1F18	1F10	GREEK CAPITAL LETTER EPSILON WITH PSILI	GREEK SMALL LETTER EPSILON WITH PSILI
1F19	1F11	GREEK CAPITAL LETTER EPSILON WITH DASIA	GREEK SMALL LETTER EPSILON WITH DASIA
1F1A	1F12	GREEK CAPITAL LETTER EPSILON WITH PSILI AND VARIA	GREEK SMALL LETTER EPSILON WITH PSILI AND VARIA
1F1B	1F13	GREEK CAPITAL LETTER EPSILON WITH DASIA AND VARIA	GREEK SMALL LETTER EPSILON WITH DASIA AND VARIA
1F1C	1F14	GREEK CAPITAL LETTER EPSILON WITH PSILI AND OXIA	GREEK SMALL LETTER EPSILON WITH PSILI AND OXIA
1F1D	1F15	GREEK CAPITAL LETTER EPSILON WITH DASIA AND OXIA	GREEK SMALL LETTER EPSILON WITH DASIA AND OXIA
1F28	1F20	GREEK CAPITAL LETTER ETA WITH PSILI	GREEK SMALL LETTER ETA WITH PSILI
1F29	1F21	GREEK CAPITAL LETTER ETA WITH DASIA	GREEK SMALL LETTER ETA WITH DASIA
1F2A	1F22	GREEK CAPITAL LETTER ETA WITH PSILI AND VARIA	GREEK SMALL LETTER ETA WITH PSILI AND VARIA
1F2B	1F23	GREEK CAPITAL LETTER ETA WITH DASIA AND VARIA	GREEK SMALL LETTER ETA WITH DASIA AND VARIA
1F2C	1F24	GREEK CAPITAL LETTER ETA WITH PSILI AND OXIA	GREEK SMALL LETTER ETA WITH PSILI AND OXIA
1F2D	1F25	GREEK CAPITAL LETTER ETA WITH DASIA AND OXIA	GREEK SMALL LETTER ETA WITH DASIA AND OXIA
1F2E	1F26	GREEK CAPITAL LETTER ETA WITH PSILI AND PERISPOMENI	GREEK SMALL LETTER ETA WITH PSILI AND PERISPOMENI
1F2F	1F27	GREEK CAPITAL LETTER ETA WITH DASIA AND PERISPOMENI	GREEK SMALL LETTER ETA WITH DASIA AND PERISPOMENI
1F38	1F30	GREEK CAPITAL LETTER IOTA WITH PSILI	GREEK SMALL LETTER IOTA WITH PSILI
1F39	1F31	GREEK CAPITAL LETTER IOTA WITH DASIA	GREEK SMALL LETTER IOTA WITH DASIA
1F3A	1F32	GREEK CAPITAL LETTER IOTA WITH PSILI AND VARIA	GREEK SMALL LETTER IOTA WITH PSILI AND VARIA
1F3B	1F33	GREEK CAPITAL LETTER IOTA WITH DASIA AND VARIA	GREEK SMALL LETTER IOTA WITH DASIA AND VARIA

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
1F3C	1F34	GREEK CAPITAL LETTER IOTA WITH PSILI AND OXIA	GREEK SMALL LETTER IOTA WITH PSILI AND OXIA
1F3D	1F35	GREEK CAPITAL LETTER IOTA WITH DASIA AND OXIA	GREEK SMALL LETTER IOTA WITH DASIA AND OXIA
1F3E	1F36	GREEK CAPITAL LETTER IOTA WITH PSILI AND PERISPOMENI	GREEK SMALL LETTER IOTA WITH PSILI AND PERISPOMENI
1F3F	1F37	GREEK CAPITAL LETTER IOTA WITH DASIA AND PERISPOMENI	GREEK SMALL LETTER IOTA WITH DASIA AND PERISPOMENI
1F48	1F40	GREEK CAPITAL LETTER OMICRON WITH PSILI	GREEK SMALL LETTER OMICRON WITH PSILI
1F49	1F41	GREEK CAPITAL LETTER OMICRON WITH DASIA	GREEK SMALL LETTER OMICRON WITH DASIA
1F4A	1F42	GREEK CAPITAL LETTER OMICRON WITH PSILI AND VARIA	GREEK SMALL LETTER OMICRON WITH PSILI AND VARIA
1F4B	1F43	GREEK CAPITAL LETTER OMICRON WITH DASIA AND VARIA	GREEK SMALL LETTER OMICRON WITH DASIA AND VARIA
1F4C	1F44	GREEK CAPITAL LETTER OMICRON WITH PSILI AND OXIA	GREEK SMALL LETTER OMICRON WITH PSILI AND OXIA
1F4D	1F45	GREEK CAPITAL LETTER OMICRON WITH DASIA AND OXIA	GREEK SMALL LETTER OMICRON WITH DASIA AND OXIA
1F59	1F51	GREEK CAPITAL LETTER UPSILON WITH OASIS	GREEK SMALL LETTER UPSILON WITH DASIA
1F5B	1F53	GREEK CAPITAL LETTER UPSILON WITH DASIA AND VARIA	GREEK SMALL LETTER UPSILON WITH DASIA AND VARIA
1F5D	1F55	GREEK CAPITAL LETTER UPSILON WITH DASIA AND OXIA	GREEK SMALL LETTER UPSILON WITH DASIA AND OXIA
1F5F	1F57	GREEK CAPITAL LETTER UPSILON WITH DASIA AND PERISPOMENI	GREEK SMALL LETTER UPSILON WITH DASIA AND PERISPOMENI
1F68	1F60	GREEK CAPITAL LETTER OMEGA WITH PSILI	GREEK SMALL LETTER OMEGA WITH PSILI
1F69	1F61	GREEK CAPITAL LETTER OMEGA WITH DASIA	GREEK SMALL LETTER OMEGA WITH DASIA
1F6A	1F62	GREEK CAPITAL LETTER OMEGA WITH PSILI AND VARIA	GREEK SMALL LETTER OMEGA WITH PSILI AND VARIA
1F6B	1F63	GREEK CAPITAL LETTER OMEGA WITH DASIA AND VARIA	GREEK SMALL LETTER OMEGA WITH DASIA AND VARIA
1F6C	1F64	GREEK CAPITAL LETTER OMEGA WITH PSILI AND OXIA	GREEK SMALL LETTER OMEGA WITH PSILI AND OXIA
1F6D	1F65	GREEK CAPITAL LETTER OMEGA WITH DASIA AND OXIA	GREEK SMALL LETTER OMEGA WITH DASIA AND OXIA

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
1F6E	1F66	GREEK CAPITAL LETTER OMEGA WITH PSILI AND PERISPOMENI	GREEK SMALL LETTER OMEGA WITH PSILI AND PERISPOMENI
1F6F	1F67	GREEK CAPITAL LETTER OMEGA WITH DASIA AND PERISPOMENI	GREEK SMALL LETTER OMEGA WITH DASIA AND PERISPOMENI
1F88	1F80	GREEK CAPITAL LETTER ALPHA WITH PSILI AND PROSGEGRAMMENI	GREEK SMALL LETTER ALPHA WITH PSILI AND YPOGEGRAMMENI
1F89	1F81	GREEK CAPITAL LETTER ALPHA WITH DASIA AND PROSGEGRAMMENI	GREEK SMALL LETTER ALPHA WITH DASIA AND YPOGEGRAMMENI
1F8A	1F82	GREEK CAPITAL LETTER ALPHA WITH PSILI AND VARIA AND PROSGEGRAMMENI	GREEK SMALL LETTER ALPHA WITH PSILI AND VARIA AND YPOGEGRAMMENI
1F8B	1F83	GREEK CAPITAL LETTER ALPHA WITH DASIA AND VARIA AND PROSGEGRAMMENI	GREEK SMALL LETTER ALPHA WITH DASIA AND VARIA AND YPOGEGRAMMENI
1F8C	1F84	GREEK CAPITAL LETTER ALPHA WITH PSILI AND OXIA AND PROSGEGRAMMEN	GREEK SMALL LETTER ALPHA WITH PSILI AND OXIA AND YPOGEGRAMMENI
1F8D	1F85	GREEK CAPITAL LETTER ALPHA WITH DASIA AND OXIA AND PROSGEGRAMMEN	GREEK SMALL LETTER ALPHA WITH DASIA AND OXIA AND YPOGEGRAMMENI
1F8E	1F86	GREEK CAPITAL LETTER ALPHA WITH PSILI AND PERISPOMENI AND PROSGEGRAMMENI	GREEK SMALL LETTER ALPHA WITH PSILI AND PERISPOMENI AND YPOGEGRAMMENI
1F8F	1F87	GREEK CAPITAL LETTER ALPHA WITH DASIA AND PERISPOMENI AND PROSGEGRAMMENI	GREEK SMALL LETTER ALPHA WITH DASIA AND PERISPOMENI AND YPOGEGRAMMENI
1F98	1F90	GREEK CAPITAL LETTER ETA WITH PSILI AND PROSGEGRAMMENI	GREEK SMALL LETTER ETA WITH PSILI AND YPOGEGRAMMENI
1F99	1F91	GREEK CAPITAL LETTER ETA WITH DASIA AND PROSGEGRAMMENI	GREEK SMALL LETTER ETA WITH DASIA AND YPOGEGRAMMENI
1F9A	1F92	GREEK CAPITAL LETTER ETA WITH PSILI AND VARIA AND PROSGEGRAMMENI	GREEK SMALL LETTER ETA WITH PSILI AND VARIA AND YPOGEGRAMMENI
1F9B	1F93	GREEK CAPITAL LETTER ETA WITH DASIA AND VARIA AND PROSGEGRAMMENI	GREEK SMALL LETTER ETA WITH DASIA AND VARIA AND YPOGEGRAMMENI
1F9C	1F94	GREEK CAPITAL LETTER ETA WITH PSILI AND OXIA AND PROSGEGRAMMENI	GREEK SMALL LETTER ETA WITH PSILI AND OXIA AND YPOGEGRAMMENI
1F9D	1F95	GREEK CAPITAL LETTER ETA WITH DASIA AND OXIA AND PROSGEGRAMMENI	GREEK SMALL LETTER ETA WITH DASIA AND OXIA AND YPOGEGRAMMENI

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
1F9E	1F96	GREEK CAPITAL LETTER ETA WITH PSILI AND PERISPOMENI AND PROSGEGRAMMENI	GREEK SMALL LETTER ETA WITH PSILI AND PERISPOMENI AND YPOGEGRAMMENI
1F9F	1F97	GREEK CAPITAL LETTER ETA WITH DASIA AND PERISPOMENI AND PROSGEGRAMMENI	GREEK SMALL LETTER ETA WITH DASIA AND PERISPOMENI AND YPOGEGRAMMENI
1FA8	1FA0	GREEK CAPITAL LETTER OMEGA WITH PSILI AND PROSGEGRAMMENI	GREEK SMALL LETTER OMEGA WITH PSILI AND YPOGEGRAMMENI
1FA9	1FA1	GREEK CAPITAL LETTER OMEGA WITH DASIA AND PROSGEGRAMMENI	GREEK SMALL LETTER OMEGA WITH DASIA AND YPOGEGRAMMENI
1FAA	1FA2	GREEK CAPITAL LETTER OMEGA WITH PSILI AND VARIA AND PROSGEGRAMMENI	GREEK SMALL LETTER OMEGA WITH PSILI AND VARIA AND YPOGEGRAMMENI
1FAB	1FA3	GREEK CAPITAL LETTER OMEGA WITH DASIA AND VARIA AND PROSGEGRAMMENI	GREEK SMALL LETTER OMEGA WITH DASIA AND VARIA AND YPOGEGRAMMENI
1FAC	1FA4	GREEK CAPITAL LETTER OMEGA WITH PSILI AND OXIA AND PROSGEGRAMMENI	GREEK SMALL LETTER OMEGA WITH PSILI AND OXIA AND YPOGEGRAMMENI
1FAD	1FA5	GREEK CAPITAL LETTER OMEGA WITH DASIA AND OXIA AND PROSGEGRAMMENI	GREEK SMALL LETTER OMEGA WITH DASIA AND OXIA AND YPOGEGRAMMENI
1FAE	1FA6	GREEK CAPITAL LETTER OMEGA WITH PSILI AND PERISPOMENI AND PROSGEGRAMMENI	GREEK SMALL LETTER OMEGA WITH PSILI AND PERISPOMENI AND YPOGEGRAMMENI
1FAF	1FA7	GREEK CAPITAL LETTER OMECA WITH DASIA AND PERISPOMENI AND PROSGEGRAMMENI	GREEK SMALL LETTER OMEGA WITH DASIA AND PEPISPOMENI AND YPOGEGRAMMENI
1FB8	1FB0	GREEK CAPITAL LETTER ALPHA WITH VRACHY	GREEK SMALL LETTER ALPHA WITH VRACHY
1FB9	1FB1	GREEK CAPITAL LETTER ALPHA WITH MACRON	GREEK SMALL LETTER ALPHA WITH MACRON
1FD8	1FD0	GREEK CAPITAL LETTER IOTA WITH VRACHY	GREEK SMALL LETTER IOTA WITH VRACHY
1FD9	1FD1	GREEK CAPITAL LETTER IOTA WITH MACRON	GREEK SMALL LETTER IOTA WITH MACRON
1FE8	1FE0	GREEK CAPITAL LETTER UPSILON WITH VRACHY	GREEK SMALL LETTER UPSILON WITH VRACHY
1FE9	1FE1	GREEK CAPITAL LETTER UPSILON WITH MACRON	GREEK SMALL LETTER UPSILON WITH MACRON
24B6	24D0	CIRCLED LATIN CAPITAL LETTER A	CIRCLED LATIN SMALL LETTER A
24B7	24D1	CIRCLED LATIN CAPITAL LETTER B	CIRCLED LATIN SMALL LETTER B

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
24B8	24D2	CIRCLED LATIN CAPITAL LETTER C	CIRCLED LATIN SMALL LETTER C
24B9	24D3	CIRCLED LATIN CAPITAL LETTER D	CIRCLED LATIN SMALL LETTER D
24BA	24D4	CIRCLED LATIN CAPITAL LETTER E	CIRCLED LATIN SMALL LETTER E
24BB	24D5	CIRCLED LATIN CAPITAL LETTER F	CIRCLED LATIN SMALL LETTER F
24BC	24D6	CIRCLED LATIN CAPITAL LETTER G	CIRCLED LATIN SMALL LETTER G
24BD	24D7	CIRCLED LATIN CAPITAL LETTER H	CIRCLED LATIN SMALL LETTER H
24BE	24D8	CIRCLED LATIN CAPITAL LETTER I	CIRCLED LATIN SMALL LETTER I
24BF	24D9	CIRCLED LATIN CAPITAL LETTER J	CIRCLED LATIN SMALL LETTER J
24C0	24DA	CIRCLED LATIN CAPITAL LETTER K	CIRCLED LATIN SMALL LETTER K
24C1	24DB	CIRCLED LATIN CAPITAL LETTER L	CIRCLED LATIN SMALL LETTER L
24C2	24DC	CIRCLED LATIN CAPITAL LETTER M	CIRCLED LATIN SMALL LETTER M
24C3	24DD	CIRCLED LATIN CAPITAL LETTER N	CIRCLED LATIN SMALL LETTER N
24C4	24DE	CIRCLED LATIN CAPITAL LETTER O	CIRCLED LATIN SMALL LETTER O
24C5	24DF	CIRCLED LATIN CAPITAL LETTER P	CIRCLED LATIN SMALL LETTER P
24C6	24E0	CIRCLED LATIN CAPITAL LETTER Q	CIRCLED LATIN SMALL LETTER Q
24C7	24E1	CIRCLED LATIN CAPITAL LETTER R	CIRCLED LATIN SMALL LETTER R
24C8	24E2	CIRCLED LATIN CAPITAL LETTER S	CIRCLED LATIN SMALL LETTER S
24C9	24E3	CIRCLED LATIN CAPITAL LETTER T	CIRCLED LATIN SMALL LETTER T
24CA	24E4	CIRCLED LATIN CAPITAL LETTER U	CIRCLED LATIN SMALL LETTER U
24CB	24E5	CIRCLED LATIN CAPITAL LETTER V	CIRCLED LATIN SMALL LETTER V
24CC	24E6	CIRCLED LATIN CAPITAL LETTER W	CIRCLED LATIN SMALL LETTER W
24CD	24E7	CIRCLED LATIN CAPITAL LETTER X	CIRCLED LATIN SMALL LETTER X
24CE	24E8	CIRCLED LATIN CAPITAL LETTER Y	CIRCLED LATIN SMALL LETTER Y

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
24CF	24E9	CIRCLED LATIN CAPITAL LETTER Z	CIRCLED LATIN SMALL LETTER Z
FF21	FF41	FULLWIDTH LATIN CAPITAL LETTER A	FULLWIDTH LATIN SMALL LETTER A
FF22	FF42	FULLWIDTH LATIN CAPITAL LETTER B	FULLWIDTH LATIN SMALL LETTER B
FF23	FF43	FULLWIDTH LATIN CAPITAL LETTER C	FULLWIDTH LATIN SMALL LETTER C
FF24	FF44	FULLWIDTH LATIN CAPITAL LETTER D	FULLWIDTH LATIN SMALL LETTER D
FF25	FF45	FULLWIDTH LATIN CAPITAL LETTER E	FULLWIDTH LATIN SMALL LETTER E
FF26	FF46	FULLWIDTH LATIN CAPITAL LETTER F	FULLWIDTH LATIN SMALL LETTER F
FF27	FF47	FULLWIDTH LATIN CAPITAL LETTER G	FULLWIDTH LATIN SMALL LETTER G
FF28	FF48	FULLWIDTH LATIN CAPITAL LETTER H	FULLWIDTH LATIN SMALL LETTER H
FF29	FF49	FULLWIDTH LATIN CAPITAL LETTER I	FULLWIDTH LATIN SMALL LETTER I
FF2A	FF4A	FULLWIDTH LATIN CAPITAL LETTER J	FULLWIDTH LATIN SMALL LETTER J
FF2B	FF4B	FULLWIDTH LATIN CAPITAL LETTER K	FULLWIDTH LATIN SMALL LETTER K
FF2C	FF4C	FULLWIDTH LATIN CAPITAL LETTER L	FULLWIDTH LATIN SMALL LETTER L
FF2D	FF4D	FULLWIDTH LATIN CAPITAL LETTER M	FULLWIDTH LATIN SMALL LETTER M
FF2E	FF4E	FULLWIDTH LATIN CAPITAL LETTER N	FULLWIDTH LATIN SMALL LETTER N
FF2F	FF4F	FULLWIDTH LATIN CAPITAL LETTER O	FULLWIDTH LATIN SMALL LETTER O
FF30	FF50	FULLWIDTH LATIN CAPITAL LETTER P	FULLWIDTH LATIN SMALL LETTER P
FF31	FF51	FULLWIDTH LATIN CAPITAL LETTER Q	FULLWIDTH LATIN SMALL LETTER Q
FF32	FF52	FULLWIDTH LATIN CAPITAL LETTER R	FULLWIDTH LATIN SMALL LETTER R
FF33	FF53	FULLWIDTH LATIN CAPITAL LETTER S	FULLWIDTH LATIN SMALL LETTER S
FF34	FF54	FULLWIDTH LATIN CAPITAL LETTER T	FULLWIDTH LATIN SMALL LETTER T
FF35	FF55	FULLWIDTH LATIN CAPITAL LETTER U	FULLWIDTH LATIN SMALL LETTER U
FF36	FF56	FULLWIDTH LATIN CAPITAL LETTER V	FULLWIDTH LATIN SMALL LETTER V

Uppercase code point	Lowercase code point	Uppercase character description	Lowercase character description
FF37	FF57	FULLWIDTH LATIN CAPITAL LETTER W	FULLWIDTH LATIN SMALL LETTER W
FF38	FF58	FULLWIDTH LATIN CAPITAL LETTER X	FULLWIDTH LATIN SMALL LETTER X
FF39	FF59	FULLWIDTH LATIN CAPITAL LETTER Y	FULLWIDTH LATIN SMALL LETTER Y
FF3A	FF5A	FULLWIDTH LATIN CAPITAL LETTER Z	FULLWIDTH LATIN SMALL LETTER Z

ISO 10646 lowercase to uppercase mapping table*:*

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
0061	0041	LATIN SMALL LETTER A	LATIN CAPITAL LETTER A
0062	0042	LATIN SMALL LETTER B	LATIN CAPITAL LETTER B
0063	0043	LATIN SMALL LETTER C	LATIN CAPITAL LETTER C
0064	0044	LATIN SMALL LETTER D	LATIN CAPITAL LETTER D
0065	0045	LATIN SMALL LETTER E	LATIN CAPITAL LETTER E
0066	0046	LATIN SMALL LETTER F	LATIN CAPITAL LETTER F
0067	0047	LATIN SMALL LETTER G	LATIN CAPITAL LETTER G
0068	0048	LATIN SMALL LETTER H	LATIN CAPITAL LETTER H
0069	0049	LATIN SMALL LETTER I	LATIN CAPITAL LETTER I
006A	004A	LATIN SMALL LETTER J	LATIN CAPITAL LETTER J
006B	004B	LATIN SMALL LETTER K	LATIN CAPITAL LETTER K
006C	004C	LATIN SMALL LETTER L	LATIN CAPITAL LETTER L
006D	004D	LATIN SMALL LETTER M	LATIN CAPITAL LETTER M
006E	004E	LATIN SMALL LETTER N	LATIN CAPITAL LETTER N
006F	004F	LATIN SMALL LETTER O	LATIN CAPITAL LETTER O
0070	0050	LATIN SMALL LETTER P	LATIN CAPITAL LETTER P
0071	0051	LATIN SMALL LETTER Q	LATIN CAPITAL LETTER Q
0072	0052	LATIN SMALL LETTER R	LATIN CAPITAL LETTER R
0073	0053	LATIN SMALL LETTER S	LATIN CAPITAL LETTER S
0074	0054	LATIN SMALL LETTER T	LATIN CAPITAL LETTER T
0075	0055	LATIN SMALL LETTER U	LATIN CAPITAL LETTER U
0076	0056	LATIN SMALL LETTER V	LATIN CAPITAL LETTER V
0077	0057	LATIN SMALL LETTER W	LATIN CAPITAL LETTER W
0078	0058	LATIN SMALL LETTER X	LATIN CAPITAL LETTER X
0079	0059	LATIN SMALL LETTER Y	LATIN CAPITAL LETTER Y
007A	005A	LATIN SMALL LETTER Z	LATIN CAPITAL LETTER Z
00E0	00C0	LATIN SMALL LETTER A GRAVE	LATIN CAPITAL LETTER A GRAVE
00E1	00C1	LATIN SMALL LETTER A GRAVE	LATIN CAPITAL LETTER A ACUTE

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
00E2	00C2	LATIN SMALL LETTER A GRAVE	LATIN CAPITAL LETTER A CIRCUMFLEX
00E3	00C3	LATIN SMALL LETTER A GRAVE	LATIN CAPITAL LETTER A TILDE
00E4	00C4	LATIN SMALL LETTER A GRAVE	LATIN CAPITAL LETTER A DIAERESIS
00E5	00C5	LATIN SMALL LETTER A GRAVE	LATIN CAPITAL LETTER A RING
00E6	00C6	LATIN SMALL LETTER A GRAVE	LATIN CAPITAL LETTER A E
00E7	00C7	LATIN SMALL LETTER A GRAVE	LATIN CAPITAL LETTER C CEDILLA
00E8	00C8	LATIN SMALL LETTER A GRAVE	LATIN CAPITAL LETTER E GRAVE
00E9	00C9	LATIN SMALL LETTER A GRAVE	LATIN CAPITAL LETTER E ACUTE
00EA	00CA	LATIN SMALL LETTER E CIRCUMFLEX	LATIN CAPITAL LETTER E CIRCUMFLEX
00EB	00CB	LATIN SMALL LETTER E DIAERESIS	LATIN CAPITAL LETTER E DIAERESIS
00EC	00CC	LATIN SMALL LETTER I GRAVE	LATIN CAPITAL LETTER I GRAVE
00ED	00CD	LATIN SMALL LETTER I ACUTE	LATIN CAPITAL LETTER I ACUTE
00EE	00CE	LATIN SMALL LETTER I CIRCUMFLEX	LATIN CAPITAL LETTER I CIRCUMFLEX
00EF	00CF	LATIN SMALL LETTER I DIAERESIS	LATIN CAPITAL LETTER I DIAERESIS
00F0	00D0	LATIN SMALL LETTER ETH	LATIN CAPITAL LETTER ETH
00F1	00D1	LATIN SMALL LETTER N TILDE	LATIN CAPITAL LETTER N TILDE
00F2	00D2	LATIN SMALL LETTER O GRAVE	LATIN CAPITAL LETTER O GRAVE
00F3	00D3	LATIN SMALL LETTER O ACUTE	LATIN CAPITAL LETTER O ACUTE
00F4	00D4	LATIN SMALL LETTER O CIRCUMFLEX	LATIN CAPITAL LETTER O CIRCUMFLEX
00F5	00D5	LATIN SMALL LETTER O TILDE	LATIN CAPITAL LETTER O TILDE
00F6	00D6	LATIN SMALL LETTER O DIAERESIS	LATIN CAPITAL LETTER O DIAERESIS
00F8	00D8	LATIN SMALL LETTER O SLASH	LATIN CAPITAL LETTER O SLASH
00F9	00D9	LATIN SMALL LETTER U GRAVE	LATIN CAPITAL LETTER U GRAVE
00FA	00DA	LATIN SMALL LETTER U ACUTE	LATIN CAPITAL LETTER U ACUTE
00FB	00DB	LATIN SMALL LETTER U CIRCUMFLEX	LATIN CAPITAL LETTER U CIRCUMFLEX
00FC	00DC	LATIN SMALL LETTER U DIAERESIS	LATIN CAPITAL LETTER U DIAERESIS
00FD	00DD	LATIN SMALL LETTER Y ACUTE	LATIN CAPITAL LETTER Y ACUTE
00FE	00DE	LATIN SMALL LETTER THORN	LATIN CAPITAL LETTER THORN
00FF	0178	LATIN SMALL LETTER Y DIAERESIS	LATIN CAPITAL LETTER Y WITH DIAERESIS
0101	0100	LATIN SMALL LETTER A WITH MACRON	LATIN CAPITAL LETTER A WITH MACRON
0103	0102	LATIN SMALL LETTER A WITH BREVE	LATIN CAPITAL LETTER A WITH BREVE

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
0105	0104	LATIN SMALL LETTER A WITH OGONEK	LATIN CAPITAL LETTER A WITH OGONEK
0107	0106	LATIN SMALL LETTER C WITH ACUTE	LATIN CAPITAL LETTER C WITH ACUTE
0109	0108	LATIN SMALL LETTER C WITH CIRCUMFLEX	LATIN CAPITAL LETTER C WITH CIRCUMFLEX
010B	010A	LATIN SMALL LETTER C WITH DOT ABOVE	LATIN CAPITAL LETTER C WITH DOT ABOVE
010D	010C	LATIN SMALL LETTER C WITH CARON	LATIN CAPITAL LETTER C WITH CARON
010F	010E	LATIN SMALL LETTER D WITH CARON	LATIN CAPITAL LETTER D WITH CARON
0111	0110	LATIN SMALL LETTER D WITH STROKE	LATIN CAPITAL LETTER D WITH STROKE
0113	0112	LATIN SMALL LETTER E WITH MACRON	LATIN CAPITAL LETTER E WITH MACRON
0115	0114	LATIN SMALL LETTER E WITH BREVE	LATIN CAPITAL LETTER E WITH BREVE
0117	0116	LATIN SMALL LETTER E WITH DOT ABOVE	LATIN CAPITAL LETTER E WITH DOT ABOVE
0119	0118	LATIN SMALL LETTER E WITH OGONEK	LATIN CAPITAL LETTER E WITH OGONEK
011B	011A	LATIN SMALL LETTER E WITH CARON	LATIN CAPITAL LETTER E WITH CARON
011D	011C	LATIN SMALL LETTER G WITH CIRCUMFLEX	LATIN CAPITAL LETTER G WITH CIRCUMFLEX
011F	011E	LATIN SMALL LETTER G WITH BREVE	LATIN CAPITAL LETTER G WITH BREVE
0121	0120	LATIN SMALL LETTER G WITH DOT ABOVE	LATIN CAPITAL LETTER G WITH DOT ABOVE
0123	0122	LATIN SMALL LETTER G WITH CEDILLA	LATIN CAPITAL LETTER G WITH CEDILLA
0125	0124	LATIN SMALL LETTER H WITH CIRCUMFLEX	LATIN CAPITAL LETTER H WITH CIRCUMFLEX
0127	0126	LATIN SMALL LETTER H WITH STROKE	LATIN CAPITAL LETTER H WITH STROKE
0129	0128	LATIN SMALL LETTER I WITH TILDE	LATIN CAPITAL LETTER I WITH TILDE
012B	012A	LATIN SMALL LETTER I WITH MACRON	LATIN CAPITAL LETTER I WITH MACRON
012D	012C	LATIN SMALL LETTER I WITH BREVE	LATIN CAPITAL LETTER I WITH BREVE
012F	012E	LATIN SMALL LETTER I WITH OGONEK	LATIN CAPITAL LETTER I WITH OGONEK
0131	0049	LATIN SMALL LETTER DOTLESS I	LATIN CAPITAL LETTER I
0133	0132	LATIN SMALL LIGATURE IJ	LATIN CAPITAL LIGATURE IJ

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
0135	0134	LATIN SMALL LETTER J WITH CIRCUMFLEX	LATIN CAPITAL LETTER J WITH CIRCUMFLEX
0137	0136	LATIN SMALL LETTER K WITH CEDILLA	LATIN CAPITAL LETTER K WITH CEDILLA
013A	0139	LATIN SMALL LETTER L WITH ACUTE	LATIN CAPITAL LETTER L WITH ACUTE
013C	013B	LATIN SMALL LETTER L WITH CEDILLA	LATIN CAPITAL LETTER L WITH CEDILLA
013E	013D	LATIN SMALL LETTER L WITH CARON	LATIN CAPITAL LETTER L WITH CARON
0140	013F	LATIN SMALL LETTER L WITH MIDDLE DOT	LATIN CAPITAL LETTER L WITH MIDDLE DOT
0142	0141	LATIN SMALL LETTER L WITH STROKE	LATIN CAPITAL LETTER L WITH STROKE
0144	0143	LATIN SMALL LETTER N WITH ACUTE	LATIN CAPITAL LETTER N WITH ACUTE
0146	0145	LATIN SMALL LETTER N WITH CEDILLA	LATIN CAPITAL LETTER N WITH CEDILLA
0148	0147	LATIN SMALL LETTER N WITH CARON	LATIN CAPITAL LETTER N WITH CARON
014B	014A	LATIN SMALL LETTER ENG (SAMI)	LATIN CAPITAL LETTER ENG (SAMI)
014D	014C	LATIN SMALL LETTER O WITH MACRON	LATIN CAPITAL LETTER O WITH MACRON
014F	014E	LATIN SMALL LETTER O WITH BREVE	LATIN CAPITAL LETTER O WITH BREVE
0151	0150	LATIN SMALL LETTER O WITH DOUBLE ACUTE	LATIN CAPITAL LETTER O WITH DOUBLE ACUTE
0153	0152	LATIN SMALL LIGATURE OE	LATIN CAPITAL LIGATURE OE
0155	0154	LATIN SMALL LETTER R WITH ACUTE	LATIN CAPITAL LETTER R WITH ACUTE
0157	0156	LATIN SMALL LETTER R WITH CEDILLA	LATIN CAPITAL LETTER R WITH CEDILLA
0159	0158	LATIN SMALL LETTER R WITH CARON	LATIN CAPITAL LETTER R WITH CARON
015B	015A	LATIN SMALL LETTER S WITH ACUTE	LATIN CAPITAL LETTER S WITH ACUTE
015D	015C	LATIN SMALL LETTER S WITH CIRCUMFLEX	LATIN CAPITAL LETTER S WITH CIRCUMFLEX
015F	015E	LATIN SMALL LETTER S WITH CEDILLA	LATIN CAPITAL LETTER S WITH CEDILLA
0161	0160	LATIN SMALL LETTER S WITH CARON	LATIN CAPITAL LETTER S WITH CARON
0163	0162	LATIN SMALL LETTER T WITH CEDILLA	LATIN CAPITAL LETTER T WITH CEDILLA
0165	0164	LATIN SMALL LETTER T WITH CARON	LATIN CAPITAL LETTER T WITH CARON

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
0167	0166	LATIN SMALL LETTER T WITH STROKE	LATIN CAPITAL LETTER T WITH STROKE
0169	0168	LATIN SMALL LETTER U WITH TILDE	LATIN CAPITAL LETTER U WITH TILDE
016B	016A	LATIN SMALL LETTER U WITH MACRON	LATIN CAPITAL LETTER U WITH MACRON
016D	016C	LATIN SMALL LETTER U WITH BREVE	LATIN CAPITAL LETTER U WITH BREVE
016F	016E	LATIN SMALL LETTER U WITH RING ABOVE	LATIN CAPITAL LETTER U WITH RING ABOVE
0171	0170	LATIN SMALL LETTER U WITH DOUBLE ACUTE	LATIN CAPITAL LETTER U WITH DOUBLE ACUTE
0173	0172	LATIN SMALL LETTER U WITH OGONEK	LATIN CAPITAL LETTER U WITH OGONEK
0175	0174	LATIN SMALL LETTER W WITH CIRCUMFLEX	LATIN CAPITAL LETTER W WITH CIRCUMFLEX
0177	0176	LATIN SMALL LETTER Y WITH CIRCUMFLEX	LATIN CAPITAL LETTER Y WITH CIRCUMFLEX
017A	0179	LATIN SMALL LETTER Z WITH ACUTE	LATIN CAPITAL LETTER Z WITH ACUTE
017C	017B	LATIN SMALL LETTER Z WITH DOT ABOVE	LATIN CAPITAL LETTER Z WITH DOT ABOVE
017E	017D	LATIN SMALL LETTER Z WITH CARON	LATIN CAPITAL LETTER Z WITH CARON
0183	0182	LATIN SMALL LETTER B WITH TOPBAR	LATIN CAPITAL LETTER B WITH TOPBAR
0185	0184	LATIN SMALL LETTER TONE SIX	LATIN CAPITAL LETTER TONE SIX
0188	0187	LATIN SMALL LETTER C WITH HOOK	LATIN CAPITAL LETTER C WITH HOOK
018C	018B	LATIN SMALL LETTER D WITH TOPBAR	LATIN CAPITAL LETTER D WITH TOPBAR
0192	0191	LATIN SMALL LETTER F WITH HOOK	LATIN CAPITAL LETTER F WITH HOOK
0199	0198	LATIN SMALL LETTER K WITH HOOK	LATIN CAPITAL LETTER K WITH HOOK
01A1	01A0	LATIN SMALL LETTER O WITH HORN	LATIN CAPITAL LETTER O WITH HORN
01A3	01A2	LATIN SMALL LETTER OI	LATIN CAPITAL LETTER OI
01A5	01A4	LATIN SMALL LETTER P WITH HOOK	LATIN CAPITAL LETTER P WITH HOOK
01A8	01A7	LATIN SMALL LETTER TONE TWO	LATIN CAPITAL LETTER TONE TWO
01AD	01AC	LATIN SMALL LETTER T WITH HOOK	LATIN CAPITAL LETTER T WITH HOOK
01B0	01AF	LATIN SMALL LETTER U WITH HORN	LATIN CAPITAL LETTER U WITH HORN

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
01B4	01B3	LATIN SMALL LETTER Y WITH HOOK	LATIN CAPITAL LETTER Y WITH HOOK
01B6	01B5	LATIN SMALL LETTER Z WITH STROKE	LATIN CAPITAL LETTER Z WITH STROKE
01B9	01B8	LATIN SMALL LETTER EZH REVERSED	LATIN CAPITAL LETTER EZH REVERSED
01BD	01BC	LATIN SMALL LETTER TONE FIVE	LATIN CAPITAL LETTER TONE FIVE
01C6	01C4	LATIN SMALL LETTER DZ WITH CARON	LATIN CAPITAL LETTER DZ WITH CARON
01C9	01C7	LATIN SMALL LETTER LJ	LATIN CAPITAL LETTER LJ
01CC	01CA	LATIN SMALL LETTER NJ	LATIN CAPITAL LETTER NJ
01CE	01CD	LATIN SMALL LETTER A WITH CARON	LATIN CAPITAL LETTER A WITH CARON
01D0	01CF	LATIN SMALL LETTER I WITH CARON	LATIN CAPITAL LETTER I WITH CARON
01D2	01D1	LATIN SMALL LETTER O WITH CARON	LATIN CAPITAL LETTER O WITH CARON
01D4	01D3	LATIN SMALL LETTER U WITH CARON	LATIN CAPITAL LETTER U WITH CARON
01D6	01D5	LATIN SMALL LETTER U WITH DIAERESIS AND MACRON	LATIN CAPITAL LETTER U WITH DIAERESIS AND MACRON
01D8	01D7	LATIN SMALL LETTER U WITH DIAERESIS AND ACUTE	LATIN CAPITAL LETTER U WITH DIAERESIS AND ACUTE
01DA	01D9	LATIN SMALL LETTER U WITH DIAERESIS AND CARON	LATIN CAPITAL LETTER U WITH DIAERESIS AND CARON
01DC	01DB	LATIN SMALL LETTER U WITH DIAERESIS AND GRAVE	LATIN CAPITAL LETTER U WITH DIAERESIS AND GRAVE
01DF	01DE	LATIN SMALL LETTER A WITH DIAERESIS AND MACRON	LATIN CAPITAL LETTER A WITH DIAERESIS AND MACRON
01E1	01E0	LATIN SMALL LETTER A WITH DOT ABOVE AND MACRON	LATIN CAPITAL LETTER A WITH DOT ABOVE AND MACRON
01E3	01E2	LATIN SMALL LIGATURE AE WITH MACRON	LATIN CAPITAL LIGATURE AE MTH MACRON
01E5	01E4	LATIN SMALL LETTER G WITH STROKE	LATIN CAPITAL LETTER G WITH STROKE
01E7	01E6	LATIN SMALL LETTER G WITH CARON	LATIN CAPITAL LETTER G WITH CARON
01E9	01E8	LATIN SMALL LETTER K WITH CARON	LATIN CAPITAL LETTER K WITH CARON
01EB	01EA	LATIN SMALL LETTER O WITH OGONEK	LATIN CAPITAL LETTER O WITH OGONEK
01ED	01EC	LATIN SMALL LETTER O WITH OGONEK AND MACRON	LATIN CAPITAL LETTER O WITH OGONEK AND MACRON
01EF	01EE	LATIN SMALL LETTER EZH WITH CARON	LATIN CAPITAL LETTER EZH WITH CARON
01F3	01F1	LATIN SMALL LETTER DZ	LATIN CAPITAL LETTER DZ

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
01F5	01F4	LATIN SMALL LETTER G WITH ACUTE	LATIN CAPITAL LETTER G WITH ACUTE
01FB	01FA	LATIN SMALL LETTER A WITH RING ABOVE AND ACUTE	LATIN CAPITAL LETTER A WITH RING ABOVE AND ACUTE
01FD	01FC	LATIN SMALL LIGATURE AE WITH ACUTE	LATIN CAPITAL LIGATURE AE WITH ACUTE
01FF	01FE	LATIN SMALL LETTER O WITH STROKE AND ACUTE	LATIN CAPITAL LETTER O WITH STROKE AND ACUTE
0201	0200	LATIN SMALL LETTER A WITH DOUBLE GRAVE	LATIN CAPITAL LETTER A WITH DOUBLE GRAVE
0203	0202	LATIN SMALL LETTER A WITH INVERTED BREVE	LATIN CAPITAL LETTER A WITH INVERTED BREVE
0205	0204	LATIN SMALL LETTER E WITH DOUBLE GRAVE	LATIN CAPITAL LETTER E WITH DOUBLE GRAVE
0207	0206	LATIN SMALL LETTER E WITH INVERTED BREVE	LATIN CAPITAL LETTER E WITH INVERTED BREVE
0209	0208	LATIN SMALL LETTER I WITH DOUBLE GRAVE	LATIN CAPITAL LETTER I WITH DOUBLE GRAVE
020B	020A	LATIN SMALL LETTER I WITH INVERTED BREVE	LATIN CAPITAL LETTER I WITH INVERTED BREVE
020D	020C	LATIN SMALL LETTER O WITH DOUBLE GRAVE	LATIN CAPITAL LETTER O WITH DOUBLE GRAVE
020F	020E	LATIN SMALL LETTER O WITH INVERTED BREVE	LATIN CAPITAL LETTER O WITH INVERTED BREVE
0211	0210	LATIN SMALL LETTER R WITH DOUBLE GRAVE	LATIN CAPITAL LETTER R WITH DOUBLE GRAVE
0213	0212	LATIN SMALL LETTER R WITH INVERTED BREVE	LATIN CAPITAL LETTER R WITH INVERTED BREVE
0215	0214	LATIN SMALL LETTER U WITH DOUBLE GRAVE	LATIN CAPITAL LETTER U WITH DOUBLE GRAVE
0217	0216	LATIN SMALL LETTER U WITH INVERTED BREVE	LATIN CAPITAL LETTER U WITH INVERTED BREVE
0253	0181	LATIN SMALL LETTER B WITH HOOK	LATIN CAPITAL LETTER B WITH HOOK
0254	0186	LATIN SMALL LETTER OPEN O	LATIN CAPITAL LETTER OPEN O
0257	018A	LATIN SMALL LETTER D WITH HOOK	LATIN CAPITAL LETTER D WITH HOOK
0258	018E	LATIN SMALL LETTER REVERSED E	LATIN CAPITAL LETTER REVERSED E
0259	018F	LATIN SMALL LETTER SCHWA	LATIN CAPITAL LETTER SCHWA
025B	0190	LATIN SMALL LETTER OPEN E	LATIN CAPITAL LETTER OPEN E
0260	0193	LATIN SMALL LETTER G WITH HOOK	LATIN CAPITAL LETTER G WITH HOOK
0263	0194	LATIN SMALL LETTER GAMMA	LATIN CAPITAL LETTER GAMMA
0268	0197	LATIN SMALL LETTER I WITH STROKE	LATIN CAPITAL LETTER I WITH STROKE

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
0269	0196	LATIN SMALL LETTER IOTA	LATIN CAPITAL LETTER IOTA
026F	019C	LATIN SMALL LETTER TURNED M	LATIN CAPITAL LETTER TURNED M
0272	019D	LATIN SMALL LETTER N WITH LEFT HOOK	LATIN CAPITAL LETTER N WITH LEFT HOOK
0275	019F	LATIN SMALL LETTER BARRED O	LATIN CAPITAL LETTER O WITH MIDDLE TILDE
0283	01A9	LATIN SMALL LETTER ESH	LATIN CAPITAL LETTER ESH
0288	01AE	LATIN SMALL LETTER T WITH RETROFLEX HOOK	LATIN CAPITAL LETTER T WITH RETROFLEX HOOK
028A	01B1	LATIN SMALL LETTER UPSILON	LATIN CAPITAL LETTER UPSILON
028B	01B2	LATIN SMALL LETTER V WITH HOOK	LATIN CAPITAL LETTER V WITH HOOK
0292	01B7	LATIN SMALL LETTER EZH	LATIN CAPITAL LETTER EZH
03AC	0386	GREEK SMALL LETTER ALPHA WITH TONOS	GREEK CAPITAL LETTER ALPHA WITH TONOS
03AD	0388	GREEK SMALL LETTER EPSILON WITH TONOS	GREEK CAPITAL LETTER EPSILON WITH TONOS
03AE	0389	GREEK SMALL LETTER ETA WITH TONOS	GREEK CAPITAL LETTER ETA WITH TONOS
03AF	038A	GREEK SMALL LETTER IOTA WITH TONOS	GREEK CAPITAL LETTER IOTA WITH TONOS
03B1	0391	GREEK SMALL LETTER ALPHA	GREEK CAPITAL LETTER ALPHA
03B2	0392	GREEK SMALL LETTER BETA	GREEK CAPITAL LETTER BETA
03B3	0393	GREEK SMALL LETTER GAMMA	GREEK CAPITAL LETTER GAMMA
03B4	0394	GREEK SMALL LETTER DELTA	GREEK CAPITAL LETTER DELTA
03B5	0395	GREEK SMALL LETTER EPSILON	GREEK CAPITAL LETTER EPSILON
03B6	0396	GREEK SMALL LETTER ZETA	GREEK CAPITAL LETTER ZETA
03B7	0397	GREEK SMALL LETTER ETA	GREEK CAPITAL LETTER ETA
03B8	0398	GREEK SMALL LETTER THETA	GREEK CAPITAL LETTER THETA
03B9	0399	GREEK SMALL LETTER IOTA	GREEK CAPITAL LETTER IOTA
03BA	039A	GREEK SMALL LETTER KAPPA	GREEK CAPITAL LETTER KAPPA
03BB	039B	GREEK SMALL LETTER LAMDA	GREEK CAPITAL LETTER LAMDA
03BC	039C	GREEK SMALL LETTER MU	GREEK CAPITAL LETTER MU
03BD	039D	GREEK SMALL LETTER NU	GREEK CAPITAL LETTER NU
03BE	039E	GREEK SMALL LETTER XI	GREEK CAPITAL LETTER XI
03BF	039F	GREEK SMALL LETTER OMICRON	GREEK CAPITAL LETTER OMICRON
03C0	03A0	GREEK SMALL LETTER PI	GREEK CAPITAL LETTER PI
03C1	03A1	GREEK SMALL LETTER RHO	GREEK CAPITAL LETTER RHO
03C3	03A3	GREEK SMALL LETTER SIGMA	GREEK CAPITAL LETTER SIGMA
03C4	03A4	GREEK SMALL LETTER TAU	GREEK CAPITAL LETTER TAU

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
03C5	03A5	GREEK SMALL LETTER UPSILON	GREEK CAPITAL LETTER UPSILON
03C6	03A6	GREEK SMALL LETTER PHI	GREEK CAPITAL LETTER PHI
03C7	03A7	GREEK SMALL LETTER CHI	GREEK CAPITAL LETTER CHI
03C8	03A8	GREEK SMALL LETTER PSI	GREEK CAPITAL LETTER PSI
03C9	03A9	GREEK SMALL LETTER OMEGA	GREEK CAPITAL LETTER OMEGA
03CA	03AA	GREEK SMALL LETTER IOTA WITH DIALYTIKA	GREEK CAPITAL LETTER IOTA WITH DIALYTIKA
03CB	03AB	GREEK SMALL LETTER UPSILON WITH DIALYTIKA	GREEK CAPITAL LETTER UPSILON WITH DIALYTIKA
03CC	038C	GREEK SMALL LETTER OMICRON WITH TONOS	GREEK CAPITAL LETTER OMICRON WITH TONOS
03CD	038E	GREEK SMALL LETTER UPSILON WITH TONOS	GREEK CAPITAL LETTER UPSILON WITH TONOS
03CE	038F	GREEK SMALL LETTER OMEGA WITH TONOS	GREEK CAPITAL LETTER OMEGA WITH TONOS
03E3	03E2	COPTIC SMALL LETTER SHEI	COPTIC CAPITAL LETTER SHEI
03E5	03E4	COPTIC SMALL LETTER FEI	COPTIC CAPITAL LETTER FEI
03E7	03E6	COPTIC SMALL LETTER KHEI	COPTIC CAPITAL LETTER KHEI
03E9	03E8	COPTIC SMALL LETTER HORI	COPTIC CAPITAL LETTER HORI
03EB	03EA	COPTIC SMALL LETTER GANGIA	COPTIC CAPITAL LETTER GANGIA
03ED	03EC	COPTIC SMALL LETTER SHIMA	COPTIC CAPITAL LETTER SHIMA
03EF	03EE	COPTIC SMALL LETTER DEI	COPTIC CAPITAL LETTER DEI
0430	0410	CYRILLIC SMALL LETTER A	CYRILLIC CAPITAL LETTER A
0431	0411	CYRILLIC SMALL LETTER BE	CYRILLIC CAPITAL LETTER BE
0432	0412	CYRILLIC SMALL LETTER VE	CYRILLIC CAPITAL LETTER VE
0433	0413	CYRILLIC SMALL LETTER GHE	CYRILLIC CAPITAL LETTER GHE
0434	0414	CYRILLIC SMALL LETTER DE	CYRILLIC CAPITAL LETTER DE
0435	0415	CYRILLIC SMALL LETTER IE	CYRILLIC CAPITAL LETTER IE
0436	0416	CYRILLIC SMALL LETTER ZHE	CYRILLIC CAPITAL LETTER ZHE
0437	0417	CYRILLIC SMALL LETTER ZE	CYRILLIC CAPITAL LETTER ZE
0438	0418	CYRILLIC SMALL LETTER I	CYRILLIC CAPITAL LETTER I
0439	0419	CYRILLIC SMALL LETTER SHORT I	CYRILLIC CAPITAL LETTER SHORT I
043A	041A	CYRILLIC SMALL LETTER KA	CYRILLIC CAPITAL LETTER KA
043B	041B	CYRILLIC SMALL LETTER EL	CYRILLIC CAPITAL LETTER EL
043C	041C	CYRILLIC SMALL LETTER EM	CYRILLIC CAPITAL LETTER EM
043D	041D	CYRILLIC SMALL LETTER EN	CYRILLIC CAPITAL LETTER EN
043E	041E	CYRILLIC SMALL LETTER O	CYRILLIC CAPITAL LETTER O
043F	041F	CYRILLIC SMALL LETTER PE	CYRILLIC CAPITAL LETTER PE
0440	0420	CYRILLIC SMALL LETTER ER	CYRILLIC CAPITAL LETTER ER
0441	0421	CYRILLIC SMALL LETTER ES	CYRILLIC CAPITAL LETTER ES

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
0442	0422	CYRILLIC SMALL LETTER TE	CYRILLIC CAPITAL LETTER TE
0443	0423	CYRILLIC SMALL LETTER U	CYRILLIC CAPITAL LETTER U
0444	0424	CYRILLIC SMALL LETTER EF	CYRILLIC CAPITAL LETTER EF
0445	0425	CYRILLIC SMALL LETTER HA	CYRILLIC CAPITAL LETTER HA
0446	0426	CYRILLIC SMALL LETTER TSE	CYRILLIC CAPITAL LETTER TSE
0447	0427	CYRILLIC SMALL LETTER CHE	CYRILLIC CAPITAL LETTER CHE
0448	0428	CYRILLIC SMALL LETTER SHA	CYRILLIC CAPITAL LETTER SHA
0449	0429	CYRILLIC SMALL LETTER SHCHA	CYRILLIC CAPITAL LETTER SHCHA
044A	042A	CYRILLIC SMALL LETTER HARD SIGN	CYRILLIC CAPITAL LETTER HARD SIGN
044B	042B	CYRILLIC SMALL LETTER YERU	CYRILLIC CAPITAL LETTER YERU
044C	042C	CYRILLIC SMALL LETTER SOFT SIGN	CYRILLIC CAPITAL LETTER SOFT SIGN
044D	042D	CYRILLIC SMALL LETTER E	CYRILLIC CAPITAL LETTER E
044E	042E	CYRILLIC SMALL LETTER YU	CYRILLIC CAPITAL LETTER YU
044F	042F	CYRILLIC SMALL LETTER YA	CYRILLIC CAPITAL LETTER YA
0451	0401	CYRILLIC SMALL LETTER IO	CYRILLIC CAPITAL LETTER IO
0452	0402	CYRILLIC SMALL LETTER DJE (SERBOCROATIAN)	CYRILLIC CAPITAL LETTER DJE (SERBOCROATIAN)
0453	0403	CYRILLIC SMALL LETTER GJE	CYRILLIC CAPITAL LETTER GJE
0454	0404	CYRILLIC SMALL LETTER UKRAINIAN IE	CYRILLIC CAPITAL LETTER UKRAINIAN IE
0455	0405	CYRILLIC SMALL LETTER DZE	CYRILLIC CAPITAL LETTER DZE
0456	0406	CYRILLIC SMALL LETTER BYELORUSSIAN-UKRAINIAN I	CYRILLIC CAPITAL LETTER BYELORUSSIAN_UKRAINIAN I
0457	0407	CYRILLIC SMALL LETTER YI (UKRANIAN)	CYRILLIC CAPITAL LETTER YI (UKRANIAN)
0458	0408	CYRILLIC SMALL LETTER JE	CYRILLIC CAPITAL LETTER JE
0459	0409	CYRILLIC SMALL LETTER LJE	CYRILLIC CAPITAL LETTER LJE
045A	040A	CYRILLIC SMALL LETTER NJE	CYRILLIC CAPITAL LETTER NJE
045B	040B	CYRILLIC SMALL LETTER TSHE (SERBOCROATIAN)	CYRILLIC CAPITAL LETTER TSHE (SERBOCROATIAN)
045C	040C	CYRILLIC SMALL LETTER KJE	CYRILLIC CAPITAL LETTER KJE
045E	040E	CYRILLIC SMALL LETTER SHORT U (BYELORUSSIAN)	CYRILLIC CAPITAL LETTER SHORT U (BYELORUSSIAN)
045F	040F	CYRILLIC SMALL LETTER DZHE	CYRILLIC CAPITAL LETTER DZHE
0461	0460	CYRILLIC SMALL LETTER OMEGA	CYRILLIC CAPITAL LETTER OMEGA
0463	0462	CYRILLIC SMALL LETTER YAT	CYRILLIC CAPITAL LETTER YAT
0465	0464	CYRILLIC SMALL LETTER IOTIFIED E	CYRILLIC CAPITAL LETTER IOTIFIED E
0467	0466	CYRILLIC SMALL LETTER LITTLE YUS	CYRILLIC CAPITAL LETTER LITTLE YUS

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
0469	0468	CYRILLIC SMALL LETTER IOTFIED LITTLE YUS	CYRILLIC CAPITAL LETTER IOTIFIED LITTLE YUS
046B	046A	CYRILLIC SMALL LETTER BIG YUS	CYRILLIC CAPITAL LETTER BIG YUS
046D	046C	CYRILLIC SMALL LETTER IOTIFIED BIG YUS	CYRILLIC CAPITAL LETTER IOTIFIED BIG YUS
046F	046E	CYRILLIC SMALL LETTER KSI	CYRILLIC CAPITAL LETTER KSI
0471	0470	CYRILLIC SMALL LETTER PSI	CYRILLIC CAPITAL LETTER PSI
0473	0472	CYRILLIC SMALL LETTER FITA	CYRILLIC CAPITAL LETTER FITA
0475	0474	CYRILLIC SMALL LETTER IZHITSA	CYRILLIC CAPITAL LETTER IZHITSA
0477	0476	CYRILLIC SMALL LETTER IZHITSA WITH DOUBLE GRAVE ACCENT	CYRILLIC CAPITAL LETTER IZHITSA WITH DOUBLE GRAVE ACCENT
0479	0478	CYRILLIC SMALL LETTER UK	CYRILLIC CAPITAL LETTER UK
047B	047A	CYRILLIC SMALL LETTER ROUND OMEGA	CYRILLIC CAPITAL LETTER ROUND OMEGA
047D	047C	CYRILLIC SMALL LETTER OMEGA WITH TITLO	CYRILLIC CAPITAL LETTER OMEGA WITH TITLO
047F	047E	CYRILLIC SMALL LETTER OT	CYRILLIC CAPITAL LETTER OT
0481	0480	CYRILLIC SMALL LETTER KOPPA	CYRILLIC CAPITAL LETTER KOPPA
0491	0490	CYRILLIC SMALL LETTER GHE WITH UPTURN	CYRILLIC CAPITAL LETTER GHE WITH UPTURN
0493	0492	CYRILLIC SMALL LETTER GHE WITH STROKE	CYRILLIC CAPITAL LETTER GHE WITH STROKE
0495	0494	CYRILLIC SMALL LETTER GHE WITH MIDDLE HOOK	CYRILLIC CAPITAL LETTER GHE WITH MIDDLE HOOK
0497	0496	CYRILLIC SMALL LETTER ZHE WITH DESCENDER	CYRILLIC CAPITAL LETTER ZHE WITH DESCENDER
0499	0498	CYRILLIC SMALL LETTER ZE WITH DESCENDER	CYRILLIC CAPITAL LETTER ZE WITH DESCENDER
049B	049A	CYRILLIC SMALL LETTER KA WITH DESCENDER	CYRILLIC CAPITAL LETTER KA WITH DESCENDER
049D	049C	CYRILLIC SMALL LETTER KA WITH VERTICAL STROKE	CYRILLIC CAPITAL LETTER KA WITH VERTICAL STROKE
049F	049E	CYRILLIC SMALL LETTER KA WITH STROKE	CYRILLIC CAPITAL LETTER KA WITH STROKE
04A1	04A0	CYRILLIC SMALL LETTER EASHKIR KA	CYRILLIC CAPITAL LETTER BASHKIR KA
04A3	04A2	CYRILLIC SMALL LETTER EN WITH DESCENOER	CYRILLIC CAPITAL LETTER EN WITH DESCENDER
04A5	04A4	CYRILLIC SMALL LIGATURE EN GHE	CYRILLIC CAPITAL LIGATURE EN GHF
04A7	04A6	CYRILLIC SMALL LETTER PE WITH MIDDLE HOOK (ABKHASIAN)	CYRILLIC CAPITAL LETTER PE WITH MIDDLE HOOK (ABKHASIAN)

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
04A9	04A8	CYRILLIC SMALL LETTER ABKHASIAN HA	CYRILLIC CAPITAL LETTER ABKHASIAN HA
04AB	04AA	CYRILLIC SMALL LETTER ES WITH DESCENDER	CYRILLIC CAPITAL LETTER ES WITH DESCENDER
04AD	04AC	CYRILLIC SMALL LETTER TE WITH DESCENDER	CYRILLIC CAPITAL LETTER TE WITH DESCENDER
04AF	04AE	CYRILLIC SMALL LETTER STRAIGHT U	CYRILLIC CAPITAL LETTER STRAIGHT U
04B1	04B0	CYRILLIC SMALL LETTER STRAIGHT U WITH STROKE	CYRILLIC CAPITAL LETTER STRAIGHT U WITH STROKE
04B3	04B2	CYRILLIC SMALL LETTER HA WITH DESCENDER	CYRILLIC CAPITAL LETTER HA WITH DESCENDER
04B5	04B4	CYRILLIC SMALL LIGATURE TE TSE (ABKHASIAN)	CYRILLIC CAPITAL LIGATURE TE TSE (ABKHASIAN)
04B7	04B6	CYRILLIC SMALL LETTER CHE WITH DESCENDER	CYRILLIC CAPITAL LETTER CHE WITH DESCENDER
04B9	04B8	CYRILLIC SMALL LETTER CHE WITH VERTICAL STROKE	CYRILLIC CAPITAL LETTER CHE WITH VERTICAL STROKE
04BB	04BA	CYRILLIC SMALL LETTER SHHA	CYRILLIC CAPITAL LETTER SHHA
04BD	04BC	CYRILLIC SMALL LETTER ABKHASIAN CHE	CYRILLIC CAPITAL LETTER ABKHASIAN CHE
04BF	04BE	CYRILLIC SMALL LETTER ABKHASIAN CHE WITH DESCENDER	CYRILLIC CAPITAL LETTER ABKHASIAN CHE WITH DESCENDER
04C2	04C1	CYRILLIC SMALL LETTER ZHE WITH BREVE	CYRILLIC CAPITAL LETTER ZHE WITH BREVE
04C4	04C3	CYRILLIC SMALL LETTER KA WITH HOOK	CYRILLIC CAPITAL LETTER KA WITH HOOK
04C8	04C7	CYRILLIC SMALL LETTER EN WITH HOOK	CYRILLIC CAPITAL LETTER EN WITH HOOK
04CC	04CB	CYRILLIC SMALL LETTER KHAKASSIAN CHE	CYRILLIC CAPITAL LETTER KHAKASSIAN CHE
04D1	04D0	CYRILLIC SMALL LETTER A WITH BREVE	CYRILLIC CAPITAL LETTER A WITH BREVE
04D3	04D2	CYRILLIC SMALL LETTER A WITH DIAERESIS	CYRILLIC CAPITAL LETTER A WITH DIAERESIS
04D5	04D4	CYRILLIC SMALL LIGATURE A IE	CYRILLIC CAPITAL LIGATURE A IE
04D7	04D6	CYRILLIC SMALL LETTER IE WITH BREVE	CYRILLIC CAPITAL LETTER IE WITH BREVE
04D9	04D8	CYRILLIC SMALL LETTER SCHWA	CYRILLIC CAPITAL LETTER SCHWA
04DB	04DA	CYRILLIC SMALL LETTER SCHWA WITH DIAERESIS	CYRILLIC CAPITAL LETTER SCHWA WITH DIAERESIS
04DD	04DC	CYRILLIC SMALL LETTER ZHE WITH DIAERESIS	CYRILLIC CAPITAL LETTER ZHE WITH DIAERESIS
04DF	04DE	CYRILLIC SMALL LETTER ZE WITH DIAERESIS	CYRILLIC CAPITAL LETTER ZE WITH DIAERESIS

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
04E1	04E0	CYRILLIC SMALL LETTER ABKHASIAN DZE	CYRILLIC CAPITAL LETTER ABKHASIAN DZE
04E3	04E2	CYRILLIC SMALL LETTER I WITH MACRON	CYRILLIC CAPITAL LETTER I WITH MACRON
04E5	04E4	CYRILLIC SMALL LETTER I WITH DIAERESIS	CYRILLIC CAPITAL LETTER I WITH DIAERESIS
04E7	04E6	CYRILLIC SMALL LETTER O WITH DIAERESIS	CYRILLIC CAPITAL LETTER O WITH DIAERESIS
04E9	04E8	CYRILLIC SMALL LETTER BARRED O	CYRILLIC CAPITAL LETTER BARRED O
04EB	04EA	CYRILLIC SMALL LETTER BARRED O WITH DIAERESIS	CYRILLIC CAPITAL LETTER BARRED O WITH DIAERESS
04EF	04EE	CYRILLIC SMALL LETTER U WITH MACRON	CYRILLIC CAPITAL LETTER U WITH MACRON
04F1	04F0	CYRILLIC SMALL LETTER U WITH DIAERESIS	CYRILLIC CAPITAL LETTER U WITH DIAERESIS
04F3	04F2	CYRILLIC SMALL LETTER U WITH DOUBLE ACUTE	CYRILLIC CAPITAL LETTER U WITH DOUBLE ACUTE
04F5	04F4	CYRILLIC SMALL LETTER CHE AITH DIAERESIS	CYRILLIC CAPITAL LETTER CHE WITH DIAERESIS
04F9	04F8	CYRILLIC SMALL LETTER YERU WITH DIAERESIS	CYRILLIC CAPITAL LETTER YERU WITH DIAERESIS
0561	0531	ARMENIAN SMALL LETTER AYB	ARMENIAN CAPITAL LETTER AYB
0562	0532	ARMENIAN SMALL LETTER BEN	ARMENIAN CAPITAL LETTER BEN
0563	0533	ARMENIAN SMALL LETTER GIM	ARMENIAN CAPITAL LETTER GIM
0564	0534	ARMENIAN SMALL LETTER DA	ARMENIAN CAPITAL LETTER DA
0565	0535	ARMENIAN SMALL LETTER ECH	ARMENIAN CAPITAL LETTER ECH
0566	0536	ARMENIAN SMALL LETTER ZA	ARMENIAN CAPITAL LETTER ZA
0567	0537	ARMENIAN SMALL LETTER EH	ARMENIAN CAPITAL LETTER EH
0568	0538	ARMENIAN SMALL LETTER ET	ARMENIAN CAPITAL LETTER ET
0569	0539	ARMENIAN SMALL LETTER TO	ARMENIAN CAPITAL LETTER TO
056A	053A	ARMENIAN SMALL LETTER ZHE	ARMENIAN CAPITAL LETTER ZHE
056B	053B	ARMENIAN SMALL LETTER INI	ARMENIAN CAPITAL LETTER INI
056C	053C	ARMENIAN SMALL LETTER LIWN	ARMENIAN CAPITAL LETTER LIWN
056D	053D	ARMENIAN SMALL LETTER XEH	ARMENIAN CAPITAL LETTER XEH
056E	053E	ARMENIAN SMALL LETTER CA	ARMENIAN CAPITAL LETTER CA
056F	053F	ARMENIAN SMALL LETTER KEN	ARMENIAN CAPITAL LETTER KEN
0570	0540	ARMENIAN SMALL LETTER HO	ARMENIAN CAPITAL LETTER HO
0571	0541	ARMENIAN SMALL LETTER JA	ARMENIAN CAPITAL LETTER JA
0572	0542	ARMENIAN SMALL LETTER GHAD	ARMENIAN CAPITAL LETTER GHAD
0573	0543	ARMENIAN SMALL LETTER CHEH	ARMENIAN CAPITAL LETTER CHEH
0574	0544	ARMENIAN SMALL LETTER MEN	ARMENIAN CAPITAL LETTER MEN

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
0575	0545	ARMENIAN SMALL LETTER YI	ARMENIAN CAPITAL LETTER YI
0576	0546	ARMENIAN SMALL LETTER NOW	ARMENIAN CAPITAL LETTER NOW
0577	0547	ARMENIAN SMALL LETTER SNA	ARMENIAN CAPITAL LETTER SHA
0578	0548	ARMENIAN SMALL LETTER VO	ARMENIAN CAPITAL LETTER VO
0579	0549	ARMENIAN SMALL LETTER CHA	ARMENIAN CAPITAL LETTER CHA
057A	054A	ARMENIAN SMALL LETTER PEH	ARMENIAN CAPITAL LETTER PEH
057B	054B	ARMENIAN SMALL LETTER JHEH	ARMENIAN CAPITAL LETTER JHEH
057C	054C	ARMENIAN SMALL LETTER RA	ARMENIAN CAPITAL LETTER RA
057D	054D	ARMENIAN SMALL LETTER SEH	ARMENIAN CAPITAL LETTER SEH
057E	054E	ARMENIAN SMALL LETTER VEW	ARMENIAN CAPITAL LETTER VEW
057F	054F	ARMENIAN SMALL LETTER TIWN	ARMENIAN CAPITAL LETTER TIWN
0580	0550	ARMENIAN SMALL LETTER REH	ARMENIAN CAPITAL LETTER REH
0581	0551	ARMENIAN SMALL LETTER CO	ARMENIAN CAPITAL LETTER CO
0582	0552	ARMENIAN SMALL LETTER YIWN	ARMENIAN CAPITAL LETTER YIWN
0583	0553	ARMENIAN SMALL LETTER PIWP	ARMENIAN CAPITAL LETTER PIWR
0584	0554	ARMENIAN SMALL LETTER KEH	ARMENIAN CAPITAL LETTER KEH
0585	0555	ARMENIAN SMALL LETTER OH	ARMENIAN CAPITAL LETTER OH
0586	0556	ARMENIAN SMALL LETTER FEH	ARMENIAN CAPITAL LETTER FEH
10D0	10A0	GEORGIAN LETTER AN	GEORGIAN CAPITAL LETTER AN (KHUTSURI)
10D1	10A1	GEORGIAN LETTER BAN	GEORGIAN CAPITAL LETTER BAN (KHUTSURI)
10D2	10A2	GEORGIAN LETTER GAN	GEORGIAN CAPITAL LETTER GAN (KHUTSURI)
10D3	10A3	GEORGIAN LETTER DON	GEORGIAN CAPITAL LETTER DON (KHUTSURI)
10D4	10A4	GEORGIAN LETTER EN	GEORGIAN CAPITAL LETTER EN (KHUTSURI)
10D5	10A5	GEORGIAN LETTER VIN	GEORGIAN CAPITAL LETTER VIN (KHUTSURI)
10D6	10A6	GEORGIAN LETTER ZEN	GEORGIAN CAPITAL LETTER ZEN (KHUTSURI)
10D7	10A7	GEORGIAN LETTER TAN	GEORGIAN CAPITAL LETTER TAN (KHUTSURI)
10D8	10A8	GEORGIAN LETTER IN	GEORGIAN CAPITAL LETTER IN (KHUTSURI)
10D9	10A9	GEORGIAN LETTER KAN	GEORGIAN CAPITAL LETTER KAN (KHUTSURI)
10DA	10AA	GEORGIAN LETTER LAS	GEORGIAN CAPITAL LETTER LAS (KHUTSURI)
10DB	10AB	GEORGIAN LETTER MAN	GEORGIAN CAPITAL LETTER MAN (KHUTSURI)

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
10DC	10AC	GEORGIAN LETTER NAR	GEORGIAN CAPITAL LETTER NAR (KHUTSURI)
10DD	10AD	GEORGIAN LETTER ON	GEORGIAN CAPITAL LETTER ON (KHUTSURI)
10DE	10AE	GEORGIAN LETTER PAR	GEORGIAN CAPITAL LETTER PAR (KHUTSURI)
10DF	10AF	GEORGIAN LETTER ZHAR	GEORGIAN CAPITAL LETTER ZHAR (KHUTSURI)
10E0	10B0	GEORGIAN LETTER RAE	GEORGIAN CAPITAL LETTER RAE (KHUTSURI)
10E1	10B1	GEORGIAN LETTER SAN	GEORGIAN CAPITAL LETTER SAN (KHUTSURI)
10E2	10B2	GEORGIAN LETTER TAR	GEORGIAN CAPITAL LETTER TAR (KHUTSURI)
10E3	10B3	GEORGIAN LETTER UN	GEORGIAN CAPITAL LETTER UN (KHUTSURI)
10E4	10B4	GEORGIAN LETTER PHAR	GEORGIAN CAPITAL LETTER PHAR (KHUTSURI)
10E5	10B5	GEORGIAN LETTER KHAR	GEORGIAN CAPITAL LETTER KHAR (KHUTSURI)
10E6	10B6	GEORGIAN LETTER GHAN	GEORGIAN CAPITAL LETTER GHAN (KHUTSURI)
10E7	10B7	GEORGIAN LETTER QAR	GEORGIAN CAPITAL LETTER QAR (KHUTSURI)
10E8	10B8	GEORGIAN LETTER SHIN	GEORGIAN CAPITAL LETTER SHIN (KHUTSURI)
10E9	10B9	GEORGIAN LETTER CHIN	GEORGIAN CAPITAL LETTER CHIN (KHUTSURI)
10EA	10BA	GEORGIAN LETTER CAN	GEORGIAN CAPITAL LETTER CAN (KHUTSURI)
10EB	10BB	GEORGIAN LETTER JIL	GEORGIAN CAPITAL LETTER JIL (KHUTSURI)
10EC	10BC	GEORGIAN LETTER CIL	GEORGIAN CAPITAL LETTER CIL (KHUTSURI)
10ED	10BD	GEORGIAN LETTER CHAR	GEORGIAN CAPITAL LETTER CHAR (KHUTSURI)
10EE	10BE	GEORGIAN LETTER XAN	GEORGIAN CAPITAL LETTER XAN (KHUTSURI)
10EF	10BF	GEORGIAN LETTER JHAN	GEORGIAN CAPITAL LETTER JHAN (KHUTSURI)
10F0	10C0	GEORGIAN LETTER HAE	GEORGIAN CAPITAL LETTER HAE (KHUTSURI)
10F1	10C1	GEORGIAN LETTER HE	GEORGIAN CAPITAL LETTER HE (KHUTSURI)
10F2	10C2	GEORGIAN LETTER HIE	GEORGIAN CAPITAL LETTER HIE (KHUTSURI)

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
10F3	10C3	GEORGIAN LETTER WE	GEORGIAN CAPITAL LETTER WE (KHUTSURI)
10F4	10C4	GEORGIAN LETTER HAR	GEORGIAN CAPITAL LETTER HAR (KHUTSURI)
10F5	10C5	GEORGIAN LETTER HOE	GEORGIAN CAPITAL LETTER HOE (KHUTSURI)
1E01	1E00	LATIN SMALL LETTER A WITH RING BELOW	LATIN CAPITAL LETTER A WITH RING BELOW
1E03	1E02	LATIN SMALL LETTER B WITH DOT ABOVE	LATIN CAPITAL LETTER B WITH DOT ABOVE
1E05	1E04	LATIN SMALL LETTER B WITH DOT BELOW	LATIN CAPITAL LETTER B WITH DOT BELOW
1E07	1E06	LATIN SMALL LETTER B WITH LINE BELOW	LATIN CAPITAL LETTER B WITH LINE BELOW
1E09	1E08	LATIN SMALL LETTER C WITH CEDILLA AND ACUTE	LATIN CAPITAL LETTER C WITH CEDILLA AND ACUTE
1E0B	1E0A	LATIN SMALL LETTER D WITH DOT ABOVE	LATIN CAPITAL LETTER D WITH DOT ABOVE
1E0D	1E0C	LATIN SMALL LETTER D WITH DOT BELOW	LATIN CAPITAL LETTER D WITH DOT BELOW
1E0F	1E0E	LATIN SMALL LETTER D WITH LINE BELOW	LATIN CAPITAL LETTER D WITH LINE BELOW
1E11	1E10	LATIN SMALL LETTER D WITH CEDILLA	LATIN CAPITAL LETTER D WITH CEDILLA
1E13	1E12	LATIN SMALL LETTER D WITH CIRCUMFLEX BELOW	LATIN CAPITAL LETTER D WITH CIRCUMFLEX BELOW
1E15	1E14	LATIN SMALL LETTER E WITH MACRON AND GRAVE	LATIN CAPITAL LETTER E WITH MACRON AND GRAVE
1E17	1E16	LATIN SMALL LETTER E WITH MACRON AND ACUTE	LATIN CAPITAL LETTER E WITH MACRON AND ACUTE
1E19	1E18	LATIN SMALL LETTER E WITH CIRCUMFLEX BELOW	LATIN CAPITAL LETTER E WITH CIRCUMFLEX BELOW
1E1B	1E1A	LATIN SMALL LETTER E WITH TILDE BELOW	LATIN CAPITAL LETTER E WITH TILDE BELOW
1E1D	1E1C	LATIN SMALL LETTER E WITH CEDILLA AND BREVE	LATIN CAPITAL LETTER E WITH CEDILLA AND BREVE
1E1F	1E1E	LATIN SMALL LETTER F WITH DOT ABOVE	LATIN CAPITAL LETTER F WITH DOT ABOVE
1E21	1E20	LATIN SMALL LETTER G WITH MACRON	LATIN CAPITAL LETTER G WITH MACRON
1E23	1E22	LATIN SMALL LETTER H WITH DOT ABOVE	LATIN CAPITAL LETTER H WITH DOT ABOVE
1E25	1E24	LATIN SMALL LETTER H WITH DOT BELOW	LATIN CAPITAL LETTER H WITH DOT BELOW
1E27	1E26	LATIN SMALL LETTER H WITH DIAERESIS	LATIN CAPITAL LETTER H WITH DIAERESIS

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
1E29	1E28	LATIN SMALL LETTER H WITH CEDILLA	LATIN CAPITAL LETTER H WITH CEDILLA
1E2B	1E2A	LATIN SMALL LETTER H WITH BREVE BELOW	LATIN CAPITAL LETTER H WITH BREVE BELOW
1E2D	1E2C	LATIN SMALL LETTER I WITH TILDE BELOW	LATIN CAPITAL LETTER I WITH TILDE BELOW
1E2F	1E2E	LATIN SMALL LETTER I WITH DIAERESIS AND ACUTE	LATIN CAPITAL LETTER I WITH DIAERESIS AND ACUTE
1E31	1E30	LATIN SMALL LETTER K WITH ACUTE	LATIN CAPITAL LETTER K WITH ACUTE
1E33	1E32	LATIN SMALL LETTER K WITH DOT BELOW	LATIN CAPITAL LETTER K WITH DOT BELOW
1E35	1E34	LATIN SMALL LETTER K WITH LINE BELOW	LATIN CAPITAL LETTER K WITH LINE BELOW
1E37	1E36	LATIN SMALL LETTER L WITH DOT BELOW	LATIN CAPITAL LETTER L WITH DOT BELOW
1E39	1E38	LATIN SMALL LETTER L WITH DOT BELOW AND MACRON	LATIN CAPITAL LETTER L WITH DOT BELOW AND MACRON
1E3B	1E3A	LATIN SMALL LETTER L WITH LINE BELOW	LATIN CAPITAL LETTER L WITH LINE BELOW
1E3D	1E3C	LATIN SMALL LETTER L WITH CIRCUMFLEX BELOW	LATIN CAPITAL LETTER L WITH CIRCUMFLEX BELOW
1E3F	1E3E	LATIN SMALL LETTER M WITH ACUTE	LATIN CAPITAL LETTER M WITH ACUTE
1E41	1E40	LATIN SMALL LETTER M WITH DOT ABOVE	LATIN CAPITAL LETTER M WITH DOT ABOVE
1E43	1E42	LATIN SMALL LETTER M WITH DOT BELOW	LATIN CAPITAL LETTER M WITH DOT BELOW
1E45	1E44	LATIN SMALL LETTER N WITH DOT ABOVE	LATIN CAPITAL LETTER N WITH DOT ABOVE
1E47	1E46	LATIN SMALL LETTER N WITH DOT BELOW	LATIN CAPITAL LETTER N WITH DOT BELOW
1E49	1E48	LATIN SMALL LETTER N WITH LINE BELOW	LATIN CAPITAL LETTER N WITH LINE BELOW
1E4B	1E4A	LATIN SMALL LETTER N WITH CIRCUMFLEX BELOW	LATIN CAPITAL LETTER N WITH CIRCUMFLEX BELOW
1E4D	1E4C	LATIN SMALL LETTER O WITH TILDE AND ACUTE	LATIN CAPITAL LETTER O WITH TILDE AND ACUTE
1E4F	1E4E	LATIN SMALL LETTER O WITH TILDE AND DIAERESIS	LATIN CAPITAL LETTER O WITH TILDE AND DIAERESIS
1E51	1E50	LATIN SMALL LETTER O WITH MACRON AND GRAVE	LATIN CAPITAL LETTER O WITH MACRON AND GRAVE
1E53	1E52	LATIN SMALL LETTER O WITH MACRON AND ACUTE	LATIN CAPITAL LETTER O WITH MACRON AND ACUTE
1E55	1E54	LATIN SMALL LETTER P WITH ACUTE	LATIN CAPITAL LETTER P WITH ACUTE

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
1E57	1E56	LATIN SMALL LETTER P WITH DOT ABOVE	LATIN CAPITAL LETTER P WITH DOT ABOVE
1E59	1E58	LATIN SMALL LETTER R WITH DOT ABOVE	LATIN CAPITAL LETTER R WITH DOT ABOVE
1E5B	1E5A	LATIN SMALL LETTER R WITH DOT BELOW	LATIN CAPITAL LETTER R WITH DOT BELOW
1E5D	1E5C	LATIN SMALL LETTER R WITH DOT BELOW AND MACRON	LATIN CAPITAL LETTER R WITH DOT BELOW AND MACRON
1E5F	1E5E	LATIN SMALL LETTER R WITH LINE BELOW	LATIN CAPITAL LETTER R WITH LINE BELOW
1E61	1E60	LATIN SMALL LETTER S WITH DOT ABOVE	LATIN CAPITAL LETTER S WITH DOT ABOVE
1E63	1E62	LATIN SMALL LETTER S WITH DOT BELOW	LATIN CAPITAL LETTER S WITH DOT BELOW
1E65	1E64	LATIN SMALL LETTER S WITH ACUTE AND DOT ABOVE	LATIN CAPITAL LETTER S WITH ACUTE AND DOT ABOVE
1E67	1E66	LATIN SMALL LETTER S WITH CARON AND DOT ABOVE	LATIN CAPITAL LETTER S WITH CARON AND DOT ABOVE
1E69	1E68	LATIN SMALL LETTER S WITH DOT BELOW AND DOT ABOVE	LATIN CAPITAL LETTER S WITH DOT BELOW AND DOT ABOVE
1E6B	1E6A	LATIN SMALL LETTER T WITH DOT ABOVE	LATIN CAPITAL LETTER T WITH DOT ABOVE
1E6D	1E6C	LATIN SMALL LETTER T WITH DOT BELOW	LATIN CAPITAL LETTER T WITH DOT BELOW
1E6F	1E6E	LATIN SMALL LETTER T WITH LINE BELOW	LATIN CAPITAL LETTER T WITH LINE BELOW
1E71	1E70	LATIN SMALL LETTER T WITH CIRCUMFLEX BELOW	LATIN CAPITAL LETTER T WITH CIRCUMFLEX BELOW
1E73	1E72	LATIN SMALL LETTER U WITH DIAERESIS BELOW	LATIN CAPITAL LETTER U WITH DIAERESIS BELOW
1E75	1E74	LATIN SMALL LETTER U WITH TILDE BELOW	LATIN CAPITAL LETTER U WITH TILDE BELOW
1E77	1E76	LATIN SMALL LETTER U WITH CIRCUMFLEX BELOW	LATIN CAPITAL LETTER U WITH CIRCUMFLEX BELOW
1E79	1E78	LATIN SMALL LETTER U WITH TILDE AND ACUTE	LATIN CAPITAL LETTER U WITH TILDE AND ACUTE
1E7B	1E7A	LATIN SMALL LETTER U WITH MACRON AND DIAERESIS	LATIN CAPITAL LETTER U WITH MACRON AND DIAERESIS
1E7D	1E7C	LATIN SMALL LETTER V WITH TILDE	LATIN CAPITAL LETTER V WITH TILDE
1E7F	1E7E	LATIN SMALL LETTER V WITH DOT BELOW	LATIN CAPITAL LETTER V WITH DOT BELOW
1E81	1E80	LATIN SMALL LETTER W WITH GRAVE	LATIN CAPITAL LETTER W WITH GRAVE
1E83	1E82	LATIN SMALL LETTER W WITH ACUTE	LATIN CAPITAL LETTER W WITH ACUTE

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
1E85	1E84	LATIN SMALL LETTER W WITH DIAERESIS	LATIN CAPITAL LETTER W WITH DIAERESIS
1E87	1E86	LATIN SMALL LETTER W WITH DOT ABOVE	LATIN CAPITAL LETTER W WITH DOT ABOVE
1E89	1E88	LATIN SMALL LETTER W WITH DOT BELOW	LATIN CAPITAL LETTER W WITH DOT BELOW
1E8B	1E8A	LATIN SMALL LETTER X WITH DOT ABOVE	LATIN CAPITAL LETTER X WITH DOT ABOVE
1E8D	1E8C	LATIN SMALL LETTER X WITH DIAERESIS	LATIN CAPITAL LETTER X5 WITH DIAERESIS
1E8F	1E8E	LATIN SMALL LETTER Y WITH DOT ABOVE	LATIN CAPITAL LETTER Y WITH DOT ABOVE
1E91	1E90	LATIN SMALL LETTER Z WITH CIRCUMFLEX	LATIN CAPITAL LETTER Z WITH CIRCUMFLEX
1E93	1E92	LATIN SMALL LETTER Z WITH DOT BELOW	LATIN CAPITAL LETTER Z WITH DOT BELOW
1E95	1E94	LATIN SMALL LETTER Z WITH LINE BELOW	LATIN CAPITAL LETTER Z WITH LINE BELOW
1EA1	1EA0	LATIN SMALL LETTER A WITH DOT BELOW	LATIN CAPITAL LETTER A WITH DOT BELOW
1EA3	1EA2	LATIN SMALL LETTER A WITH HOOK ABOVE	LATIN CAPITAL LETTER A WITH HOOK ABOVE
1EA5	1EA4	LATIN SMALL LETTER A WITH CIRCUMFLEX AND ACUTE	LATIN CAPITAL LETTER A WITH CIRCUMFLEX AND ACUTE
1EA7	1EA6	LATIN SMALL LETTER A WITH CIRCUMFLEX AND GRAVE	LATIN CAPITAL LETTER A WITH CIRCUMFLEX AND GRAVE
1EA9	1EA8	LATIN SMALL LETTER A WITH CIRCUMFLEX AND HOOK ABOVE	LATIN CAPITAL LETTER A WITH CIRCUMFLEX AND HOOK ABOVE
1EAB	1EAA	LATIN SMALL LETTER A WITH CIRCUMFLEX AND TILDE	LATIN CAPITAL LETTER A WITH CIRCUMFLEX AND TILDE
1EAD	1EAC	LATIN SMALL LETTER A WITH CIRCUMFLEX AND DOT BELOW	LATIN CAPITAL LETTER A WITH CIRCUMFLEX AND DOT BELOW
1EAF	1EAE	LATIN SMALL LETTER A WITH BREVE AND ACUTE	LATIN CAPITAL LETTER A WITH BREVE AND ACUTE
1EB1	1EB0	LATIN SMALL LETTER A WITH BREVE AND GRAVE	LATIN CAPITAL LETTER A WITH BREVE AND GRAVE
1EB3	1EB2	LATIN SMALL LETTER A WITH BREVE AND HOOK ABOVE	LATIN CAPITAL LETTER A WITH BREVE AND HOOK ABOVE
1EB5	1EB4	LATIN SMALL LETTER A WITH BREVE AND TILDE	LATIN CAPITAL LETTER A WITH BREVE AND TILDE
1EB7	1EB6	LATIN SMALL LETTER A WITH BREVE AND DOT BELOW	LATIN CAPITAL LETTER A WITH BREVE AND DOT BELOW
1EB9	1EB8	LATIN SMALL LETTER E WITH DOT BELOW	LATIN CAPITAL LETTER E WITH DOT BELOW
1EBB	1EBA	LATIN SMALL LETTER E WITH HOOK ABOVE	LATIN CAPITAL LETTER E WITH HOOK ABOVE

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
1EBD	1EBC	LATIN SMALL LETTER E WITH TILDE	LATIN CAPITAL LETTER E WITH TILDE
1EBF	1EBE	LATIN SMALL LETTER E WITH CIRCUMFLEX AND ACUTE	LATIN CAPITAL LETTER E WITH CIRCUMFLEX AND ACUTE
1EC1	1EC0	LATIN SMALL LETTER E WITH CIRCUMFLEX AND GRAVE	LATIN CAPITAL LETTER E WITH CIRCUMFLEX AND GRAVE
1EC3	1EC2	LATIN SMALL LETTER E WITH CIRCUMFLEX AND HOOK ABOVE	LATIN CAPITAL LETTER E WITH CIRCUMFLEX AND HOOK ABOVE
1EC5	1EC4	LATIN SMALL LETTER E WITH CIRCUMFLEX AND TILDE	LATIN CAPITAL LETTER E WITH CIRCUMFLEX AND TILDE
1EC7	1EC6	LATIN SMALL LETTER E WITH CIRCUMFLEX AND DOT BELOW	LATIN CAPITAL LETTER E WITH CIRCUMFLEX AND DOT BELOW
1EC9	1EC8	LATIN SMALL LETTER I WITH HOOK ABOVE	LATIN CAPITAL LETTER I WITH HOOK ABOVE
1ECB	1ECA	LATIN SMALL LETTER I WITH DOT BELOW	LATIN CAPITAL LETTER I WITH DOT BELOW
1ECD	1ECC	LATIN SMALL LETTER O WITH DOT BELOW	LATIN CAPITAL LETTER O WITH DOT BELOW
1ECF	1ECE	LATIN SMALL LETTER O WITH HOOK ABOVE	LATIN CAPITAL LETTER O WITH HOOK ABOVE
1ED1	1ED0	LATIN SMALL LETTER O WITH CIRCUMFLEX AND ACUTE	LATIN CAPITAL LETTER O WITH CRCUMFLEX AND ACUTE
1ED3	1ED2	LATIN SMALL LETTER O WITH CIRCUMFLEX AND GRAVE	LATIN CAPITAL LETTER O WITH CIRCUMFLEX AND GRAVE
1ED5	1ED4	LATIN SMALL LETTER O WITH CIRCUMFLEX AND HOOK ABOVE	LATIN CAPITAL LETTER O WITH CIRCUMFLEX AND HOOK ABOVE
1ED7	1ED6	LATIN SMALL LETTER O WITH CIRCUMFLEX AND TILDE	LATIN CAPITAL LETTER O WITH CIRCUMFLEX AND TILDE
1ED9	1ED8	LATIN SMALL LETTER O WITH CIRCUMFLEX AND DOT BELOW	LATIN CAPITAL LETTER O WITH CIRCUMFLEX AND DOT BELOW
1EDB	1EDA	LATIN SMALL LETTER O WITH HORN AND ACUTE	LATIN CAPITAL LETTER O WITH HORN AND ACUTE
1EDD	1EDC	LATIN SMALL LETTER O WITH HORN AND GRAVE	LATIN CAPITAL LETTER O WITH HORN AND GRAVE
1EDF	1EDE	LATIN SMALL LETTER O WITH HORN AND HOOK ABOVE	LATIN CAPITAL LETTER O WITH HORN AND HOOK ABOVE
1EE1	1EE0	LATIN SMALL LETTER O WITH HORN AND TILDE	LATIN CAPITAL LETTER O WITH HORN AND TILDE
1EE3	1EE2	LATIN SMALL LETTER O WITH HORN AND DOT BELOW	LATIN CAPITAL LETTER O WITH HORN AND DOT BELOW
1EE5	1EE4	LATIN SMALL LETTER U WITH DOT BELOW	LATIN CAPITAL LETTER U WITH DOT BELOW
1EE7	1EE6	LATIN SMALL LETTER U WITH HOOK ABOVE	LATIN CAPITAL LETTER U WITH HOOK ABOVE
1EE9	1EE8	LATIN SMALL LETTER U WITH HORN AND ACUTE	LATIN CAPITAL LETTER U WITH HORN AND ACUTE

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
1EEB	1EEA	LATIN SMALL LETTER U WITH HORN AND GRAVE	LATIN CAPITAL LETTER U WITH HORN AND GRAVE
1EED	1EEC	LATIN SMALL LETTER U WITH HORN AND HOCK ABOVE	LATIN CAPITAL LETTER U WITH HORN AND HOOK ABOVE
1EEF	1EEE	LATIN SMALL LETTER U WITH HORN AND TILDE	LATIN CAPITAL LETTER U WITH HORN AND TILDE
1EF1	1EF0	LATIN SMALL LETTER U WITH HORN AND DOT BELOW	LATIN CAPITAL LETTER U WITH HORN AND DOT BELOW
1EF3	1EF2	LATIN SMALL LETTER Y WITH GRAVE	LATIN CAPITAL LETTER Y WITH GRAVE
1EF5	1EF4	LATIN SMALL LETTER Y WITH DOT BELOW	LATIN CAPITAL LETTER Y WITH DOT BELOW
1EF7	1EF6	LATIN SMALL LETTER Y WITH HOOK ABOVE	LATIN CAPITAL LETTER Y WITH HOOK ABOVE
1EF9	1EF8	LATIN SMALL LETTER Y WITH TILDE	LATIN CAPITAL LETTER Y WITH TILDE
1F00	1F08	GREEK SMALL LETTER ALPHA WITH PSILI	GREEK CAPITAL LETTER ALPHA WITH PSILI
1F01	1F09	GREEK SMALL LETTER ALPHA WITH DASIA	GREEK CAPITAL LETTER ALPHA WITH DASIA
1F02	1F0A	GREEK SMALL LETTER ALPHA WITH PSILI AND VARIA	GREEK CAPITAL LETTER ALPHA WITH PSILI AND VARIA
1F03	1F0B	GREEK SMALL LETTER ALPHA WITH DASIA AND VARIA	GREEK CAPITAL LETTER ALPHA WITH DASIA AND VARIA
1F04	1F0C	GREEK SMALL LETTER ALPHA WITH PSILI AND OXIA	GREEK CAPITAL LETTER ALPHA WITH PSILI AND OXIA
1F05	1F0D	GREEK SMALL LETTER ALPHA WITH DASIA AND OXIA	GREEK CAPITAL LETTER ALPHA WITH DASIA AND OXIA
1F06	1F0E	GREEK SMALL LETTER ALPHA WITH PSILI AND PERISPOMENI	GREEK CAPITAL LETTER ALPHA WITH PSILI AND PERISPOMENI
1F07	1F0F	GREEK SMALL LETTER ALPHA WITH DASIA AND PERISPOMENI	GREEK CAPITAL LETTER ALPHA WITH DASIA AND PERISPOMENI
1F10	1F18	GREEK SMALL LETTER EPSILON WITH PSILI	GREEK CAPITAL LETTER EPSILON WITH PSILI
1F11	1F19	GREEK SMALL LETTER EPSILON WITH DASIA	GREEK CAPITAL LETTER EPSILON WITH DASIA
1F12	1F1A	GREEK SMALL LETTER EPSILON WITH PSILI AND VARIA	GREEK CAPITAL LETTER EPSILON WITH PSILI AND VARIA
1F13	1F1B	GREEK SMALL LETTER EPSILON WITH DASIA AND VARIA	GREEK CAPITAL LETTER EPSILON WITH DASIA AND VARIA
1F14	1F1C	GREEK SMALL LETTER EPSILON WITH PSILI AND OXIA	GREEK CAPITAL LETTER EPSILON WITH PSILI AND OXIA
1F15	1F1D	GREEK SMALL LETTER EPSILON WITH DASIA AND OXIA	GREEK CAPITAL LETTER EPSILON WITH DASIA AND OXIA
1F20	1F28	GREEK SMALL LETTER ETA WITH PSILI	GREEK CAPITAL LETTER ETA WITH PSILI

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
1F21	1F29	GREEK SMALL LETTER ETA WITH DASIA	GREEK CAPITAL LETTER ETA WITH DASIA
1F22	1F2A	GREEK SMALL LETTER ETA WITH PSILI AND VARIA	GREEK CAPITAL LETTER ETA WITH PSILI AND VARIA
1F23	1F2B	GREEK SMALL LETTER ETA WITH DASIA AND VARIA	GREEK CAPITAL LETTER ETA WITH DASIA AND VARIA
1F24	1F2C	GREEK SMALL LETTER ETA WITH PSILI AND OXIA	GREEK CAPITAL LETTER ETA WITH PSILI AND OXIA
1F25	1F2D	GREEK SMALL LETTER ETA WITH DASIA AND OXIA	GREEK CAPITAL LETTER ETA WITH DASIA AND OXIA
1F26	1F2E	GREEK SMALL LETTER ETA WITH PSILI AND PERISPOMENI	GREEK CAPITAL LETTER ETA WITH PSILI AND PERISPOMENI
1F27	1F2F	GREEK SMALL LETTER ETA WITH DASIA AND PERISPOMENI	GREEK CAPITAL LETTER ETA WITH DASIA AND PERISPOMENI
1F30	1F38	GREEK SMALL LETTER IOTA WITH PSILI	GREEK CAPITAL LETTER IOTA WITH PSILI
1F31	1F39	GREEK SMALL LETTER IOTA WITH DASIA	GREEK CAPITAL LETTER IOTA WITH DASIA
1F32	1F3A	GREEK SMALL LETTER IOTA WITH PSILI AND VARIA	GREEK CAPITAL LETTER IOTA WITH PSILI AND VARIA
1F33	1F3B	GREEK SMALL LETTER IOTA WITH DASIA AND VARIA	GREEK CAPITAL LETTER IOTA WITH DASIA AND VARIA
1F34	1F3C	GREEK SMALL LETTER IOTA WITH PSILI AND OXIA	GREEK CAPITAL LETTER IOTA WITH PSILI AND OXIA
1F35	1F3D	GREEK SMALL LETTER IOTA WITH DASIA AND OXIA	GREEK CAPITAL LETTER IOTA WITH DASIA AND OXIA
1F36	1F3E	GREEK SMALL LETTER IOTA WITH PSILI AND PERISPOMENI	GREEK CAPITAL LETTER IOTA WITH PSILI AND PERISPOMENI
1F37	1F3F	GREEK SMALL LETTER IOTA WITH DASIA AND PERISPOMENI	GREEK CAPITAL LETTER IOTA WITH DASIA AND PERISPOMENI
1F40	1F48	GREEK SMALL LETTER OMICRON WITH PSILI	GREEK CAPITAL LETTER OMICRON WITH PSILI
1F41	1F49	GREEK SMALL LETTER OMICRON WITH DASIA	GREEK CAPITAL LETTER OMICRON WITH DASIA
1F42	1F4A	GREEK SMALL LETTER OMICRON WITH PSILI AND VARIA	GREEK CAPITAL LETTER OMICRON WITH PSILI AND VARIA
1F43	1F4B	GREEK SMALL LETTER OMICRON WITH DASIA AND VARIA	GREEK CAPITAL LETTER OMICRON WITH DASIA AND VARIA
1F44	1F4C	GREEK SMALL LETTER OMICRON WITH PSILI AND OXIA	GREEK CAPITAL LETTER OMICRON WITH PSILI AND OXIA
1F45	1F4D	GREEK SMALL LETTER OMICRON WITH DASIA AND OXIA	GREEK CAPITAL LETTER OMICRON WITH DASIA AND OXIA
1F51	1F59	GREEK SMALL LETTER UPSILON WITH DASIA	GREEK CAPITAL LETTER UPSILON WITH OASIS

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
1F53	1F5B	GREEK SMALL LETTER UPSILON WITH DASIA AND VARIA	GREEK CAPITAL LETTER UPSILON WITH DASIA AND VARIA
1F55	1F5D	GREEK SMALL LETTER UPSILON WITH DASIA AND OXIA	GREEK CAPITAL LETTER UPSILON WITH DASIA AND OXIA
1F57	1F5F	GREEK SMALL LETTER UPSILON WITH DASIA AND PERISPOMENI	GREEK CAPITAL LETTER UPSILON WITH DASIA AND PERISPOMENI
1F60	1F68	GREEK SMALL LETTER OMEGA WITh PSILI	GREEK CAPITAL LETTER OMEGA WITH PSILI
1F61	1F69	GREEK SMALL LETTER OMEGA WITH DASIA	GREEK CAPITAL LETTER OMEGA WITH DASIA
1F62	1F6A	GREEK SMALL LETTER OMEGA WITH PSILI AND VARIA	GREEK CAPITAL LETTER OMEGA WITH PSILI AND VARIA
1F63	1F6B	GREEK SMALL LETTER OMEGA WITH DASIA AND VARIA	GREEK CAPITAL LETTER OMEGA WITH DASIA AND VARIA
1F64	1F6C	GREEK SMALL LETTER OMEGA WITH PSILI AND OXIA	GREEK CAPITAL LETTER OMEGA WITH PSILI AND OXIA
1F65	1F6D	GREEK SMALL LETTER OMEGA WITH DASIA AND OXIA	GREEK CAPITAL LETTER OMEGA WITH DASIA AND OXIA
1F66	1F6E	GREEK SMALL LETTER OMEGA WITH PSILI AND PERISPOMENI	GREEK CAPITAL LETTER OMEGA WITH PSILI AND PERISPOMENI
1F67	1F6F	GREEK SMALL LETTER OMEGA WITH DASIA AND PERISPOMENI	GREEK CAPITAL LETTER OMEGA WITH DASIA AND PERISPOMENI
1F80	1F88	GREEK SMALL LETTER ALPHA WITH PSILI AND YPOGEGRAMMENI	GREEK CAPITAL LETTER ALPHA WITh PSILI AND PROSGEGRAMMENI
1F81	1F89	GREEK SMALL LETTER ALPHA WITH DASIA AND YPOGEGRAMMENI	GREEK CAPITAL LETTER ALPHA WITH DASIA AND PROSGEGRAMMENI
1F82	1F8A	GREEK SMALL LETTER ALPHA WITH PSILI AND VARIA AND YPOGEGRAMMENI	GREEK CAPITAL LETTER ALPHA WITH PSILI AND VARIA AND PROSGEGRAMMENI
1F83	1F8B	GREEK SMALL LETTER ALPHA WITH DASIA AND VARIA AND YPOGEGRAMMENI	GREEK CAPITAL LETTER ALPHA WITH DASIA AND VARIA AND PROSGEGRAMMENI
1F84	1F8C	GREEK SMALL LETTER ALPHA WITH PSILI AND OXIA AND YPOGEGRAMMENI	GREEK CAPITAL LETTER ALPHA WITH PSILI AND OXIA AND PROSGEGRAMMEN
1F85	1F8D	GREEK SMALL LETTER ALPHA WITH DASIA AND OXIA AND YPOGEGRAMMENI	GREEK CAPITAL LETTER ALPHA WITH DASIA AND OXIA AND PROSGEGRAMMEN
1F86	1F8E	GREEK SMALL LETTER ALPHA WITH PSILI AND PERISPOMENI AND YPOGEGRAMMENI	GREEK CAPITAL LETTER ALPHA WITH PSILI AND PERISPOMENI AND PROSGEGRAMMENI
1F87	1F8F	GREEK SMALL LETTER ALPHA WITH DASIA AND PERISPOMENI AND YPOGEGRAMMENI	GREEK CAPITAL LETTER ALPHA WITH DASIA AND PERISPOMENI AND PROSGEGRAMMENI

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
1F90	1F98	GREEK SMALL LETTER ETA WITH PSILI AND YPOGEGRAMMENI	GREEK CAPITAL LETTER ETA WITH PSILI AND PROSGEGRAMMENI
1F91	1F99	GREEK SMALL LETTER ETA WITH DASIA AND YPOGEGRAMMENI	GREEK CAPITAL LETTER ETA WITH DASIA AND PROSGEGRAMMENI
1F92	1F9A	GREEK SMALL LETTER ETA WITH PSILI AND VARIA AND YPOGEGRAMMENI	GREEK CAPITAL LETTER ETA WITH PSILI AND VARIA AND PROSGEGRAMMENI
1F93	1F9B	GREEK SMALL LETTER ETA WITH DASIA AND VARIA AND YPOGEGRAMMENI	GREEK CAPITAL LETTER ETA WITH DASIA AND VARIA AND PROSGEGRAMMENI
1F94	1F9C	GREEK SMALL LETTER ETA WITH PSILI AND OXIA AND YPOGEGRAMMENI	GREEK CAPITAL LETTER ETA WITH PSILI AND OXIA AND PROSGEGRAMMENI
1F95	1F9D	GREEK SMALL LETTER ETA WITH DASIA AND OXIA AND YPOGEGRAMMENI	GREEK CAPITAL LETTER ETA WITH DASIA AND OXIA AND PROSGEGRAMMENI
1F96	1F9E	GREEK SMALL LETTER ETA WITH PSILI AND PERISPOMENI AND YPOGEGRAMMENI	GREEK CAPITAL LETTER ETA WITH PSILI AND PERISPOMENI AND PROSGEGRAMMENI
1F97	1F9F	GREEK SMALL LETTER ETA WITH DASIA AND PERISPOMENI AND YPOGEGRAMMENI	GREEK CAPITAL LETTER ETA WITH DASIA AND PERISPOMENI AND PROSGEGRAMMENI
1FA0	1FA8	GREEK SMALL LETTER OMEGA WITH PSILI AND YPOGEGRAMMENI	GREEK CAPITAL LETTER OMEGA WITH PSILI AND PROSGEGRAMMENI
1FA1	1FA9	GREEK SMALL LETTER OMEGA WITH DASIA AND YPOGEGRAMMENI	GREEK CAPITAL LETTER OMEGA WITH DASIA AND PROSGEGRAMMENI
1FA2	1FAA	GREEK SMALL LETTER OMEGA WITH PSILI AND VARIA AND YPOGEGRAMMENI	GREEK CAPITAL LETTER OMEGA WITH PSILI AND VARIA AND PROSGEGRAMMENI
1FA3	1FAB	GREEK SMALL LETTER OMEGA WITH DASIA AND VARIA AND YPOGEGRAMMENI	GREEK CAPITAL LETTER OMEGA WITH DASIA AND VARIA AND PROSGEGRAMMENI
1FA4	1FAC	GREEK SMALL LETTER OMEGA WITH PSILI AND OXIA AND YPOGEGRAMMENI	GREEK CAPITAL LETTER OMEGA WITH PSILI AND OXIA AND PROSGEGRAMMENI
1FA5	1FAD	GREEK SMALL LETTER OMEGA WITH DASIA AND OXIA AND YPOGEGRAMMENI	GREEK CAPITAL LETTER OMEGA WITH DASIA AND OXIA AND PROSGEGRAMMENI
1FA6	1FAE	GREEK SMALL LETTER OMEGA WITh PSILI AND PERISPOMENI AND YPOGEGRAMMENI	GREEK CAPITAL LETTER OMEGA WITH PSILI AND PERISPOMENI AND PROSGEGRAMMENI
1FA7	1FAF	GREEK SMALL LETTER OMEGA WITH DASIA AND PEPISPOMENI AND YPOGEGRAMMENI	GREEK CAPITAL LETTER OMECA WITH DASIA AND PERISPOMENI AND PROSGEGRAMMENI
1FB0	1FB8	GREEK SMALL LETTER ALPHA WITH VRACHY	GREEK CAPITAL LETTER ALPHA WITH VRACHY

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
1FB1	1FB9	GREEK SMALL LETTER ALPHA WITH MACRON	GREEK CAPITAL LETTER ALPHA WITH MACRON
1FD0	1FD8	GREEK SMALL LETTER IOTA WITH VRACHY	GREEK CAPITAL LETTER IOTA WITH VRACHY
1FD1	1FD9	GREEK SMALL LETTER IOTA WITH MACRON	GREEK CAPITAL LETTER IOTA WITH MACRON
1FE0	1FE8	GREEK SMALL LETTER UPSILON WITH VRACHY	GREEK CAPITAL LETTER UPSILON WITH VRACHY
1FE1	1FE9	GREEK SMALL LETTER UPSILON WITH MACRON	GREEK CAPITAL LETTER UPSILON WITH MACRON
24D0	24B6	CIRCLED LATIN SMALL LETTER A	CIRCLED LATIN CAPITAL LETTER A
24D1	24B7	CIRCLED LATIN SMALL LETTER B	CIRCLED LATIN CAPITAL LETTER B
24D2	24B8	CIRCLED LATIN SMALL LETTER C	CIRCLED LATIN CAPITAL LETTER C
24D3	24B9	CIRCLED LATIN SMALL LETTER D	CIRCLED LATIN CAPITAL LETTER D
24D4	24BA	CIRCLED LATIN SMALL LETTER E	CIRCLED LATIN CAPITAL LETTER E
24D5	24BB	CIRCLED LATIN SMALL LETTER F	CIRCLED LATIN CAPITAL LETTER F
24D6	24BC	CIRCLED LATIN SMALL LETTER G	CIRCLED LATIN CAPITAL LETTER G
24D7	24BD	CIRCLED LATIN SMALL LETTER H	CIRCLED LATIN CAPITAL LETTER H
24D8	24BE	CIRCLED LATIN SMALL LETTER I	CIRCLED LATIN CAPITAL LETTER I
24D9	24BF	CIRCLED LATIN SMALL LETTER J	CIRCLED LATIN CAPITAL LETTER J
24DA	24C0	CIRCLED LATIN SMALL LETTER K	CIRCLED LATIN CAPITAL LETTER K
24DB	24C1	CIRCLED LATIN SMALL LETTER L	CIRCLED LATIN CAPITAL LETTER L
24DC	24C2	CIRCLED LATIN SMALL LETTER M	CIRCLED LATIN CAPITAL LETTER M
24DD	24C3	CIRCLED LATIN SMALL LETTER N	CIRCLED LATIN CAPITAL LETTER N
24DE	24C4	CIRCLED LATIN SMALL LETTER O	CIRCLED LATIN CAPITAL LETTER O
24DF	24C5	CIRCLED LATIN SMALL LETTER P	CIRCLED LATIN CAPITAL LETTER P
24E0	24C6	CIRCLED LATIN SMALL LETTER Q	CIRCLED LATIN CAPITAL LETTER Q
24E1	24C7	CIRCLED LATIN SMALL LETTER R	CIRCLED LATIN CAPITAL LETTER R
24E2	24C8	CIRCLED LATIN SMALL LETTER S	CIRCLED LATIN CAPITAL LETTER S

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
24E3	24C9	CIRCLED LATIN SMALL LETTER T	CIRCLED LATIN CAPITAL LETTER T
24E4	24CA	CIRCLED LATIN SMALL LETTER U	CIRCLED LATIN CAPITAL LETTER U
24E5	24CB	CIRCLED LATIN SMALL LETTER V	CIRCLED LATIN CAPITAL LETTER V
24E6	24CC	CIRCLED LATIN SMALL LETTER W	CIRCLED LATIN CAPITAL LETTER W
24E7	24CD	CIRCLED LATIN SMALL LETTER X	CIRCLED LATIN CAPITAL LETTER X
24E8	24CE	CIRCLED LATIN SMALL LETTER Y	CIRCLED LATIN CAPITAL LETTER Y
24E9	24CF	CIRCLED LATIN SMALL LETTER Z	CIRCLED LATIN CAPITAL LETTER Z
FF41	FF21	FULLWIDTH LATIN SMALL LETTER A	FULLWIDTH LATIN CAPITAL LETTER A
FF42	FF22	FULLWIDTH LATIN SMALL LETTER B	FULLWIDTH LATIN CAPITAL LETTER B
FF43	FF23	FULLWIDTH LATIN SMALL LETTER C	FULLWIDTH LATIN CAPITAL LETTER C
FF44	FF24	FULLWIDTH LATIN SMALL LETTER D	FULLWIDTH LATIN CAPITAL LETTER D
FF45	FF25	FULLWIDTH LATIN SMALL LETTER E	FULLWIDTH LATIN CAPITAL LETTER E
FF46	FF26	FULLWIDTH LATIN SMALL LETTER F	FULLWIDTH LATIN CAPITAL LETTER F
FF47	FF27	FULLWIDTH LATIN SMALL LETTER G	FULLWIDTH LATIN CAPITAL LETTER G
FF48	FF28	FULLWIDTH LATIN SMALL LETTER H	FULLWIDTH LATIN CAPITAL LETTER H
FF49	FF29	FULLWIDTH LATIN SMALL LETTER I	FULLWIDTH LATIN CAPITAL LETTER I
FF4A	FF2A	FULLWIDTH LATIN SMALL LETTER J	FULLWIDTH LATIN CAPITAL LETTER J
FF4B	FF2B	FULLWIDTH LATIN SMALL LETTER K	FULLWIDTH LATIN CAPITAL LETTER K
FF4C	FF2C	FULLWIDTH LATIN SMALL LETTER L	FULLWIDTH LATIN CAPITAL LETTER L
FF4D	FF2D	FULLWIDTH LATIN SMALL LETTER M	FULLWIDTH LATIN CAPITAL LETTER M
FF4E	FF2E	FULLWIDTH LATIN SMALL LETTER N	FULLWIDTH LATIN CAPITAL LETTER N
FF4F	FF2F	FULLWIDTH LATIN SMALL LETTER O	FULLWIDTH LATIN CAPITAL LETTER O
FF50	FF30	FULLWIDTH LATIN SMALL LETTER P	FULLWIDTH LATIN CAPITAL LETTER P

Lowercase code point	Uppercase code point	Lowercase character description	Uppercase character description
FF51	FF31	FULLWIDTH LATIN SMALL LETTER Q	FULLWIDTH LATIN CAPITAL LETTER Q
FF52	FF32	FULLWIDTH LATIN SMALL LETTER R	FULLWIDTH LATIN CAPITAL LETTER R
FF53	FF33	FULLWIDTH LATIN SMALL LETTER S	FULLWIDTH LATIN CAPITAL LETTER S
FF54	FF34	FULLWIDTH LATIN SMALL LETTER T	FULLWIDTH LATIN CAPITAL LETTER T
FF55	FF35	FULLWIDTH LATIN SMALL LETTER U	FULLWIDTH LATIN CAPITAL LETTER U
FF56	FF36	FULLWIDTH LATIN SMALL LETTER V	FULLWIDTH LATIN CAPITAL LETTER V
FF57	FF37	FULLWIDTH LATIN SMALL LETTER W	FULLWIDTH LATIN CAPITAL LETTER W
FF58	FF38	FULLWIDTH LATIN SMALL LETTER X	FULLWIDTH LATIN CAPITAL LETTER X
FF59	FF39	FULLWIDTH LATIN SMALL LETTER Y	FULLWIDTH LATIN CAPITAL LETTER Y
FF5A	FF3A	FULLWIDTH LATIN SMALL LETTER Z	FULLWIDTH LATIN CAPITAL LETTER Z

GB18030: The Chinese standard

GB 18030-2000 is a Chinese standard that specifies an extended code page for use in the Chinese market. This code page standard is important for the software industry because the China National Information Technology Standardization Technical Committee has mandated that any software application that is released for the Chinese market after September 1, 2001, be enabled for GB18030.

OS/400 supports this encoding with CCSID 1392. Generally, you should use Unicode instead of 1392 for complete national language support. CCSID 1392 is provided if you need to handle or interchange GB18030 encoded data. See Unicode for more information.

A brief history of major GB code pages

A common base code page standard for Chinese is GB 2312-1980. It encodes more than 6,000 frequently-used Chinese ideographs. With the growing importance of Unicode and the parallel standard ISO 10646 (which was adopted by China as GB 13000), an extension of GB 2312-1980 was created. This extension was called GBK; it encoded all 20,902 unified ideographs that are assigned in Unicode 2.1. GBK is not a formal standard, but a widely-implemented specification.

Unicode 3.0 added more than 6,000 ideographs, and version 3.1 added about 42,000 additional ideographs.

GB 18030 was created as an update of GBK for Unicode 3.0 with an extension that covers all of Unicode. It has the following general features:

- GB 18030 character assignments are backwards compatible with the GB 2312-1980 standard and the GBK specification.

- The mapping table between GB 18030 and Unicode is backwards compatible with the one between GB 2312-1980 and Unicode, and with some exceptions (with the one between GBK and Unicode), most of the changes compared to the GBK mapping table are due to updates for Unicode 3.0.
- GB 18030 specifies a mapping table that covers all Unicode code points. It is functionally similar to a UTF (Unicode Transformation Format) while maintaining compatibility of GB-encoded text with GBK and GB 2312-1980.

Work with CCSIDs

This topic describes how the server implements the Character Data Representation Architecture (CDRA). Using the server implementation of CDRA, you can achieve consistent representation, processing, and interchange of coded characters (data) on OS/400 and across IBM systems. The primary implementation of CDRA on OS/400 is through coded character set identifier (CCSID) support.

- Recommendations and guidelines for using CCSIDs
- OS/400 function support for CCSIDs
- Change the CCSID of a physical file
- Graphic character (data) sort implementation
- CCSID support for messages

Recommendations and guidelines for using CCSIDs

When writing global applications, the following are some recommendations to remember:

- Because the system is shipped with a default CCSID of 65535, character data conversions do not normally occur in applications. You should look over the CCSID information in this topic, however, because the system may need to participate in a multilingual environment, a network, or exchanging data at a later time.
- Applications implementing their own mapping scheme should use CCSID 65535, where a CCSID assignment is necessary. For example, depending on what an application does, it might need to use CCSID 65535 for the files, or it might need to use CCSID 65535 for the jobs. Because other applications may require CCSIDs other than 65535, consider changing such applications by replacing the mapping scheme with CCSID support.
- Correctly define fields based on their usage. If fields contain application-dependent values (for example, control characters or fields that are not used as real character fields), define the fields as hexadecimal data or character fields with CCSID 65535.
- Avoid using characters that are not in the invariant character set for names and literals in programs.

Follow these guidelines when using CCSIDs:

- Use CCSIDs in multilingual applications to maintain character integrity in database files, displays, and printed data.
- You can find a suggested CCSID for a language in Language identifiers and associated default CCSIDs.
- If the QIGC system value is set on, set QCCSID as a mixed CCSID or 65535. For more information on QIGC, see DBCS system indicator (QIGC) system value.
- If you use DBCS support, set the job CCSID to a mixed CCSID. If you do not, set the job CCSID to a single-byte CCSID.
- Ensure that the QCHRID code page is compatible with the character set and code page of the QCCSID value, unless the QCCSID value is 65535. If the QCCSID value is changed to a value that is incompatible with the current QCHRID value, the QCHRID value is changed to a compatible value by the system.
- If you use a user-defined data stream (UDDS), remove any X'3F' values inserted by CCSID conversions. Otherwise, your data can cause the system to blank out a screen. Some CCSID conversions use a X'3F' value for a substitution character.

- If you are using any interactive jobs, such as Application Development ToolSet/400, ensure that the code page of the job CCSID matches the code page of the keyboard type. If these CCSID values do not match, or the job CCSID is 65535, unpredictable results could occur. For more information, see National language keyboard types and SBCS code pages.
- Be aware that the *JOBCCSID support is not used by any system-supplied displays or panel groups, although CHRIDCTL support is used.
- Be aware of character data that has been defined or specified as control information. For new database files, fields that contain control information should be defined as hexadecimal datatype or use CCSID 65535 instead of another CCSID.
- Because of workstation hardware restrictions, you may not see all of the characters on displays other than 3486, 3487, 3488, or Personal System/2 (PS/2(R)) displays when CCSID conversion occurs. However, the character data is retained in the system.
- Be aware that when a CCSID conversion is performed, substitution characters may cause a loss of data. The situation occurs if enforced subset match conversion is performed (see Conversion of character data).

OS/400 function support for CCSIDs

The server provides support for CCSIDs in the functions as shown in the following table:

Function	Description of support
CL commands	Some control language (CL) commands have internal functions that support CCSID conversions. For more information about CL commands that support CCSID conversions, see the CL Reference topic.
Copy	Coded character set identifier (CCSID) support is built into the copy function. The Copy File (CPYF) and Copy from Query File (CPYFRMQRYF) commands support CCSIDs. To use the CPYF command to change a physical file, see Changing the CCSID of a physical file. The Copy Source File (CPYSRCF) command supports CCSID conversion.
Database management	Database management support provides default coded character set identifier (CCSID) values for database files on the server. See the Database management topic for details.

Function	Description of support
DDM	Coded character set identifier (CCSID) support is built into distributed data management (DDM). DDM provides support to pass CCSID tags in homogeneous environments. DDM passes a CCSID parameter when sending files. With DDM, you can also specify a CCSID when creating files on a remote system. DDM only converts data to the job CCSID of the source system when: • The source and target systems are iSeries servers. • The source and target systems are at an operating system level of Version 2 Release 1.1 or later. Program-described files are always created with a CCSID of 65535 if they are created: • On a target AS/400(R) system on a release level from Version 2 Release 1.1 through Version 2 Release 3 • From a source system that is not an iSeries server • From a source system that is an AS/400 system at a release level before Version 2 Release 1.1 You can use the Submit Remote Command (SBMRMTCMD) command on a source iSeries server to change the file CCSID (externally described files only) by specifying the CHGPF command and the CCSID parameter.
DDS	Coded character set identifier (CCSID) support is built into data description specifications (DDS). DDS supports file-level and field-level CCSID keywords for all character fields in physical files. DDS also supports file-level and field-level keywords for all DBCS fields in physical files.
Distributed relational database	Coded character set identifier (CCSID) support is built into distributed relational database. Distributed relational database passes the CCSID of an application requester (AR) job to an application server (AS) job and vice versa during connect processing. Distributed relational database also performs a conversion of error information and text-describing fields according to the job CCSID. Distributed relational database uses CCSID information to determine how to build data exchanged between application requester jobs and application server jobs. It also uses CCSID information to describe data exchanged between application requester jobs and application server jobs (for example, a format description).
IDDU	Coded character set identifier (CCSID) support is built into interactive data definition utility (IDDU). Interactive data definition utility provides support to specify a CCSID for a character field or a DBCS field.

Function	Description of support
Open Query File (OPNQRYF)	Coded character set identifier (CCSID) support is built into OS/400 query. You can use the Open Query File (OPNQRYF) command to specify a CCSID on the MAPFLD parameter. The MAPFLD parameter specifies the definition of query fields that are either mapped to, or derived from, other fields. OS/400 query supports CCSID conversion on CHAR, OPEN, EITHER, and UCS-2 graphic field operators for join, record selection, group-by, and minimum or maximum values functions. CCSID conversion is performed whenever fields do not have the same CCSID value. Once the query is opened, database management support converts data read or written to the database files as described in the “Database management” topic. OS/400 query does not support CCSID conversion if at least one of the fields is assigned a CCSID of 65535.
Query management	Coded character set identifier (CCSID) support is built into query management. Query management assigns a CCSID to queries and forms. Query management: • Converts queries to the job CCSID. • Presents data to the display device using the job CCSID. • Assigns a CCSID to the files it creates.
SNA	Coded character set identifier (CCSID) support is built into SNA Distributed Services (SNADS). SNADS supports CCSIDs by any user ID, system name, or destination queue name. However, other SNADS services such as SNDNETF do not provide CCSID conversion.
Work management	Work management support provides the function to assign or change coded character set identifier (CCSID) values at three different levels. See the Work management topic for details.
Workstations	The workstation management function provides support for display files, printer files, and panel groups. See the Workstation function management topic for details.

Database management: Database management support provides default coded character set identifier (CCSID) values for database files on the server. All database files are assigned a CCSID. At file creation time, the CCSID is either explicitly assigned through DDS, SQL, or IDDU, or it is implicitly assigned the job default CCSID (DFTCCSID).

Database files support for CCSIDs:

IBM system files and licensed program database files are created with the CCSID of choice for each of the national language versions. Only the customer files are automatically assigned the CCSID of the job creating the file. You can use the Display File Description (DSPFD) command to view the CCSID of a file.

Program-described files are assigned CCSID 65535. If a CCSID is not explicitly specified on the CRTPF or CRTSRCPF command, database source files default to the job default CCSID at file creation. For more information on job default CCSIDs, see Job default coded character set identifier (DFTCCSID). For a list of language identifiers and the DFTCCSID valued associated with those identifiers, see Language Identifiers and Associated Default CCSIDs table.

If a database logical file is defined over several physical files, it is assigned a CCSID at the field level and assumes the CCSID value of the physical file. Logical files cannot be explicitly assigned a CCSID value.

Database fields and support for CCSIDs:

Except for numeric database fields, database fields are supported by CCSIDs. You can use the Display File Field Description (DSPFFD) command to view the CCSID of the fields in a file.

Hexadecimal fields are assigned CCSID 65535.

An implicit CCSID value is assigned to the following fields if a CCSID was not explicitly assigned through DDS, SQL, or IDDU at file creation:

- Physical-file character
- DBCS-open
- DBCS-only
- DBCS-either
- Graphic

The implicitly assigned CCSID is the job default CCSID, or a CCSID associated with the job default CCSID.

- A character field will be assigned the single-byte character set (SBCS) CCSID that is associated with the job default CCSID.
- A DBCS-open, DBCS-only, and DBCS-either field will be assigned the mixed byte CCSID.
- A Graphic field will be assigned the double-byte character set (DBCS) CCSID that is associated with the job default CCSID.

For example, if the job default CCSID is 5026 (which is a CCSID that identifies mixed data), an SBCS character field will be assigned the SBCS CCSID associated with 5026. Thus, the CCSID for that field would be 290. If there is no CCSID of the required character set type then a CCSID of 65535 will be used. See Job Default Coded Character Set Identifier (DFTCCSID) for more information on job default CCSIDs. See Language identifiers and associated default CCSIDS for a list of language identifiers and the DFTCCSID valued associated with those identifiers.

Database logical-file fields are assigned a CCSID value based on their data type and the data type of the underlying physical file field.

Database management and conversion support for CCSIDs:

Database management support converts non-graphic character data read from, or written to, database files using the file CCSID and the job CCSID.

- If data is being read from a database file and the CCSID of the file is the same as the job CCSID, no conversion is done.
- If data is being read from a database file and the CCSID of the file and the job CCSID are different, the data is converted to the CCSID of the job.
- If data is being written to a database file and the CCSID of the file is the same as the job CCSID, no conversion is done.
- If data is being written to a database file and the CCSID of the file and the job CCSID are different, the data is converted to match the CCSID of the file.

No conversion is performed if either the CCSID of the job or the CCSID of the database file is equal to 65535.

Work management: Work management support provides the function to assign or change coded character set identifier (CCSID) values at three different levels. All jobs run with a CCSID value established at one of these levels:

- *Job level.* A CCSID is assigned to a job.
- *User profile level.* A CCSID is specified in a user profile and the value is assigned to all jobs run under that user profile. The CCSID can be set or changed with the Create User Profile (CRTUSRPRF) and Change User Profile (CHGUSRPRF) commands.
- *System level.* The system value QCCSID is the default CCSID for all jobs running on the system. QCCSID can be set or changed with the CHGSYSVAL and WRKSYSVAL commands.

Work management support initializes the job CCSID for an interactive job to the CCSID on the user profile when the job starts. If *SYSVAL is specified for the CCSID on the user profile, work management support gets the CCSID from the system value (QCCSID). For batch jobs, the CCSID of the current job is used as the default CCSID for the submitted job.

You can change the CCSID of a job by using the Change Job (CHGJOB) command. Make a note of the current job CCSID. You can use it later to reset the job CCSID to its original value, if necessary. The new CCSID value is reflected in the job immediately. The job DFTCCSID cannot be changed. To retrieve the CCSID or DFTCCSID for a job, use the Retrieve Job Attributes (RTVJOBA) command or the Retrieve Job Information QUSRJOBI application programming interface (API). Interactively, use the Work with Job (WRKJOB) command and select the Display Job Definition Attributes option on the Work with Job display.

Workstation function management: Workstation function management involves working with:

- Display files
- Printer files
- Panel groups

All source files on the system are tagged with a coded character set identifier (CCSID).

Display files: When a display file object is created, it is tagged with the coded character set identifier (CCSID) of the source file. At compile time:

- All character data is read from the primary source file without any character conversion being performed.
- User message text (identified by the MSGCON keyword in DDS) remains the same because it is assumed to be in the same CCSID as the primary source file.

At run time, the constant data is converted based on the CHRID parameter value used to create the display file object. This conversion is optional and can occur only when the CHRID is set to *JOBCCSID or indirectly with CHRIDCTL. This conversion is from the display file CCSID to the character identifier (CHRID) of the device. The field-level keyword NOCCSID (no coded character set identifier) allows the user to specify fields within the DDS that are never to be converted.

Note: To use data management support of CCSIDs, you must change source physical files tagged with CCSID 65535 to a CCSID value that is associated with the data. See Changing the CCSID of a physical file for more information.

CHRID parameter on the Create Display File command

The CHRID parameter on the Create Display File (CRTDSPF) command affects the conversion that occurs for the display file.

If the *JOBCCSID value is specified on the CHRID parameter of the CRTDSPF command:

- Input characters are converted from the device character identifier (CHRID) to the job CCSID.
- Character data is sent to output-capable fields and converted from the job CCSID to the device CHRID.

- Constant text from the display file is converted from the CCSID of the display file to the CHRID of the device.
- All message files are tagged with a CCSID. Message text is converted from the CCSID of the message file to the CHRID of the device. When message files are tagged with a CCSID of 65535 (the system default), it is assumed that the contents of the message files are already in the CHRID of the device. To ensure appropriate conversions occur, you can enable CCSID support for messages. See CCSID message support for more information on enabling CCSID support for messages.
- Message replacement data is converted from the CCSID of the job, or from the CCSID of the display file, to the CHRID of the device.
- All status messages that are tagged with a CCSID other than 65535 are converted to the CHRID of the device.
- Message text for messages on a message line or in a message subfile (identified by the ERRMSG, ERRMSGID, SFLMSG, and SFLMSGID keywords in DDS) is converted from the message file CCSID to the device CHRID.

If a specific value is specified for the CHRID parameter on the CRTDSPF command, conversion is done between the CHRID specified on the CRTDSPF command and the CHRID of the device. This conversion affects only fields defined with the CHRID DDS keyword.

If the *DEVD value is specified on the CHRID parameter of the CRTDSPF command, no conversion is performed. This is the default setting.

Migration of display files with CCSID 65535

All source files in Version 3 of the OS/400 licensed program have an implicit CCSID value of 65535. To have appropriate CCSID support, display files must be recompiled with a source file that has a CCSID value other than 65535 if either of the following are true:

- The display file was originally compiled from a source file with a CCSID value of 65535.
- The display file was originally compiled prior to Version 2 Release 3 Modification 0 of the OS/400 licensed program.

By recompiling, the display file object is tagged and all necessary conversions take place when needed.

No conversions take place if the source files are explicitly tagged CCSID 65535.

Printer files: When a printer file object is created, it is tagged with the coded character set identifier (CCSID) of the source file. Processing of the source files for printer files is the same as for display files. At compile time, all character data is read from the primary source file without any character conversion being performed.

When printing to the device, if the *JOBCCSID value is specified on the CHRID parameter of the CRTPRTF command:

- Constant text from an externally described printer file is converted from the CCSID of the printer file to the CCSID of the job.
- Character data sent to output fields is assumed to be already converted to the job CCSID.

If the printer data stream is tagged with the character identifier (CHRID) derived from the CCSID of the job, the CHRID value is used by the printer to interpret the data. The CHRID value is ignored for printers not supporting this function.

If a specific value is set for the CHRID parameter on the CRTPRTF command:

- For externally described printer files, fields that specify the CHRID DDS keyword use the CHRID value specified on the printer file. The remainder of the file is printed as if *DEVD was specified for the CHRID parameter on the CRTPRTF command.

- For program-described printer files, the printer data stream uses the CHRID value specified on the printer file.

If the *DEVD parameter is specified on the CHRID parameter of the CRTPRTF command, no conversion is performed.

The CHRID information is determined by either the printer hardware or by the device description. If the CHRID information is obtained from the device description, it is then sent to the printer.

User interface manager menus and panel groups: Like display files and printer files, panel group objects and user interface manager (UIM) menus are tagged with the CCSID of the primary source file. The contents of embedded source members are converted to this CCSID. When the panel group or UIM menu is created with *JOBCCSID specified for the CHRID parameter, conversion is performed at run time. Conversion is performed between the CCSIDs of the panel group or menu, the job, and the CHRID of the display or printer.

CCSID conversions of user interface manager menu and panel groups

The following CCSID conversions occur for displays of panel groups and UIM menus:

- Text in the panel group is converted from the panel group CCSID to the device CHRID.
- Text in the UIM menu is converted from the UIM menu CCSID to the CHRID of the device.
- Variables from the user job are converted from the job CCSID to the device CHRID.
- Variables from the job are converted from the CHRID of the device to the job CCSID.
- Online help information imported from a different panel group is converted from the imported panel group CCSID to the device CHRID.

CCSID conversions when printing UIM menus and panel groups

CCSID conversions for printed UIM menus and panel groups are shown in the following table. In this table, *xxx* and *yyy* are explicitly assigned CCSID values. For example, a printer file CHRID is explicitly assigned a value of 00697 00037. The panel group is set to *JOBCCSID. The panel group constant text is converted from the panel group primary source file tagged with CCSID 00500 to the printer file CHRID 00697 00037.

Printer file CHRID is	**And the panel group or menu CCSID is xxx**	**or *JOBCCSID**	**or *DEVD**
yyy	No conversion occurs for panel group constant text.	Panel group constant text is converted from panel group primary source file CCSID to yyy.	No conversion occurs for panel group constant text.
	Variables with CHRID=PNLGRP on class tag are converted from xxx to yyy.	Variables with CHRID=PNLGRP on class tag are converted from job CCSID to yyy.	No conversion occurs for variables with CHRID=PNLGRP on class tag.
	No conversion occurs for variables without CHRID=PNLGRP on class tag.	Variables without CHRID=PNLGRP on class tag are converted from job CCSID to YYY.	No conversion occurs for variables without CHRID=PNLGRP on class tag.
*JOBCCSID	No conversion occurs for panel group constant text.	Panel group constant text is converted from panel group primary source file CCSID to job CCSID.	Panel group constant text is converted from panel group primary source file CCSID to job CCSID.

Printer file CHRID is	And the panel group or menu CCSID is xxx	or *JOBCCSID	or *DEVD
	Variables with CHRID=PNLGRP on class tag are converted from XXX to job CCSID.	No conversion occurs for variables with CHRID=PNLGRP on class tag.	No conversion occurs for variables with CHRID=PNLGRP on class tag.
	No conversion occurs for variables without CHRID=PNLGRP on class tag.	No conversion occurs for variables without CHRID=PNLGRP on class tag.	No conversion occurs for variables without CHRID=PNLGRP on class tag.
*DEVD	No conversion occurs for panel group constant text.	Panel group constant text is converted from panel group primary source file CCSID to job CCSID. This conversion occurs because variables are in the job CCSID and the device CHRID is unknown.	No conversion occurs for panel group constant text.
	No conversion occurs for variables with CHRID=PNLGRP on class tag.	No conversion occurs for variables with CHRID=PNLGRP on class tag.	No conversion occurs for variables with CHRID=PNLGRP on class tag.
	No conversion occurs for variables without CHRID=PNLGRP on class tag.	No conversion occurs for variables without CHRID=PNLGRP on class tag.	No conversion occurs for variables without CHRID=PNLGRP on class tag.

Change the CCSID of a physical file

You can use the Change Physical File (CHGPF) command to change the coded character set identifier (CCSID) of a physical file.

However, a physical file cannot be changed if one or more of the following conditions exist when working with a logical file defined over a physical file:

- The logical file has a sort sequence table associated with the CCSID of the physical file and the CCSID you want to change to is incompatible. That is, conversion between the original CCSID and the CCSID you want to change to is not allowed because all the characters of the original CCSID are not in the new CCSID.
- The logical file has a sort sequence table associated with the CCSID of the physical file and the CCSID you want to change to is incompatible. Additionally, the logical file has fields defined with CCSIDs that are not compatible to the new CCSID you want to change the physical file to. Again, conversion between the original CCSID and the CCSID you want to change to is not allowed because all the characters of the original CCSID of the logical file or the fields with specific CCSIDs are not in the new CCSID.
- A select/omit or join logical file, or both that performs select/omits or joins between physical file fields that have different CCSIDs.
- A join logical file with a sort sequence table such that the CCSID of the logical file's secondary access path is different than the CCSID to which the physical file is being changed.

Graphic character (data) sort implementation

The following links describe the OS/400 implementation of sorting, or sequencing, characters (data). The server allows you to customize the sequence in which single-byte and graphic characters are sorted. You can customize the sorting sequence of a set of characters using a sort sequence table.

If your application uses locales, you can use the sorting support provided by the LC_COLLATE locale category.

Use the following links to find additional information about character graphic sort implementation:
- Sort sequence types
- Sort sequence scenarios
- Sort sequence support
- Sort sequence tables

For more information

See Sort sequences.

Sort sequence types: A set of shared-weight and unique-weight sort sequence tables for SBCS languages is provided on servers. A shared-weight sequence is a sort sequence in which some graphic characters may have the same weight as some other characters in the sequence. Those with the same weight sort together as though they were the same character. For example, the letters *a* and *A* might both have the same value 24. This ensures that words such as *able* and *Able* are kept together in a list. In a simple sort table, *a* and *A* might share the value 24, and *b* and *B* might share the value 25 and so on.

A unique-weight sequence is a sort sequence in which each graphic character has a weight different from the weight of every other graphic character in the sequence.

Sort sequence scenarios: The following table shows characters you can sort using a binary, a shared-weight, and a unique-weight sort sequence for the Danish code page 00277.

Character name	Character illustration	Code point in code page 277	Shared sort weight	Unique sort weight
AE ligature	Æ	X'7B'	96	183
O slash	Ø	X'7C'	97	187
A overcircle	Å	X'5B'	98	191
Latin capital N	N	X'D5'	83	132
Latin capital Z	Z	X'E9'	95	181
O umlaut	Ö	X'EC'	97	189
Latin capital A	A	X'C1'	70	77

Using the information in the previous table, the characters are sorted in ascending order as shown in the following table.

Position in ascending order	Binary sort	Shared weight sort	Unique weight sort
First	A overcircle	Latin capital A	Latin capital A
Second	AE ligature	Latin capital N	Latin capital N
Third	O slash	Latin capital Z	Latin capital Z
Fourth	Latin capital A	AE ligature	AE ligature

Position in ascending order	Binary sort	Shared weight sort	Unique weight sort
Fifth	Latin capital N	O umlaut	O slash
Sixth	Latin capital Z	O slash	O umlaut
Seventh	O umlaut	A overcircle	A overcircle

The following table shows an example of a shared-weight sort sequence, a unique weight sort sequence, and the binary sort sequence for English code page 00037.

Binary sort sequence	Shared-weight sort sequence using LANGID(ENU) and SRTSEQ(*LANGIDSHR)	Unique-weight sort sequence using LANGID(ENU) and SRTSEQ(*LANGIDUNQ)
Jones, Mary	JOHNSON, JOHN	JOHNSON, JOHN
JOHNSON, JOHN	JONES, MARTIN	Jones, Mary
JONES, MARTIN	Jones, Mary	JONES, MARTIN
Smith, Ron	SMITH, ROBERT	Smith, Ron
SMITH, ROBERT	Smith, Ron	SMITH, ROBERT

Sort sequence support: The sort sequence support is provided in the following OS/400 functions.

- A user interface for creating new tables based on system-supplied sort sequence tables
- The Work with Tables (WRKTBL) command for creating and displaying tables
- The Create Table (CRTTBL) command for creating tables
- CL, ILE RPG IV, and ILE COBOL for compilers.
- Program support
- Work management support
- Database management support
- Other system components support

Sort sequence support in programs: You can assign sort sequences to programs used for ordering and comparing data. You assign a sort sequence to a program by specifying the sort sequence to be used at compile time. Specify the sort sequence to be used with the sort sequence (SRTSEQ) parameter and language identifier (LANGID) parameters of the create program commands. Valid SRTSEQ parameter values are:

- SRTSEQ(*HEX) means that no sort sequence should be used (hexadecimal sorting).
- SRTSEQ(*LANGIDUNQ) or SRTSEQ(*LANGIDSHR) means that the unique- or shared-weight sort sequence, determined by the LANGID parameter, should be used.
- A name for the system-supplied or user-supplied sort sequence name can be specified explicitly on the SRTSEQ parameter. If you explicitly specify a sort sequence name, the LANGID parameter is ignored.
- SRTSEQ(*JOB) or LANGID(*JOB) means that the sort sequence to be used is determined by the value associated with the job when the program is created.
- SRTSEQ(*JOBRUN) or LANGID(*JOBRUN) means that the sort sequence to be used is determined by the values from the job when the program is run.

The first three options assign the sort sequence to the program object at creation time. This sequence is always used when the program is run. Using the *JOBRUN value on the SRTSEQ or LANGID parameters, however, provides the possibility for dynamically assigning sort sequence to the program.

Sort sequence support in work management: Work management involves the assigning of the SRTSEQ value at the job level, the user profile level, and the system value level.

Sort sequence support at the job level: A sort sequence (SRTSEQ) value is assigned to a job. It is valid on the Submit Job (SBMJOB), Batch Job (BCHJOB), and the Change Job (CHGJOB) commands. If a program is created with SRTSEQ(*JOB), the sort sequence is set from the job sort sequence. If a program is created with SRTSEQ(*JOBRUN), the sort sequence is set from the job sort sequence at run time.

Sort sequence support at the user profile level: The user profile assigns a SRTSEQ value to a user and, by default, to all jobs running under this user profile. The user profile SRTSEQ value defaults to the sort sequence system value (QSRTSEQ).

Sort sequence support at the system value level: The QSRTSEQ system value defines a sort sequence that can be referred to by other objects. The QSRTSEQ system value should be set according to the requirements of the primary language used on the system. For more information on QSRTSEQ, see Sort Sequence (QSRTSEQ) system value.

Sort sequence support in database management: Database management supports the SRTSEQ and LANGID parameters on the Create Physical File (CRTPF) and Create Logical File (CRTLF) commands.

The LANGID and SRTSEQ parameters determine a sort sequence table. The sort sequence table is captured at file creation time and is stored as an attribute of the file. The SRTSEQ job attribute has no effect on the processing of an existing database file. The sort sequence table associated with the file is used for key sequencing, select logic fields and omit logic fields, and for join field functions.

The ALTSEQ keyword in DDS can also be used to specify a sort sequence table. The ALTSEQ keyword applies only to the key fields, not to the select logic fields and the omit logic fields. If the SRTSEQ parameter is specified on the CRTPF command or the CRTLF commands and the ALTSEQ keyword in the DDS source file specify a sort sequence table, an error message is sent and the file is not created.

The default SRTSEQ parameter on CRTPF and CRTLF commands is *SRC, which indicates that the sort sequence table on the ALTSEQ keyword should be used. If ALTSEQ is not used in DDS, the SRTSEQ attribute of the job determines the file attributes when creating or changing the file.

How sort sequences are specified for database management

Sort sequence tables can be specified in the following areas:

- Query for iSeries support

  External sort sequence tables (including those shipped with the system) and user-defined tables can be specified.
- DB2 Query Manager and SQL Development Kit for iSeries

  The Create Structured Query Language xxx (CRTSQLxxx) commands and the Start Structured Query Language (STRSQL) command support the SRTSEQ and LANGID parameters.

  A sort sequence table can be specified when a query object is being defined with the Work with Queries display. The sort sequence (SRTSEQ) value and language identifier (LANGID) value are specified on the Specify Sort Sequence display.
- DB2 UDB for iSeries Query Management

  The Create Query Management Query (CRTQMQRY) command supports the SRTSEQ and LANGID parameters.

For more information on sort sequence support for database programming, see the DB2 UDB for Database Programming topic.

Sort sequence support in other system components: Sort sequence support is found in the following components of the system:

- CRTCLPGM (Create Control Language Program) command

  The LANGID and SRTSEQ parameters are supported.

- DSPPGM (Display Program) command

  The LANGID and SRTSEQ values that were specified when the program was created are displayed.
- CRTDSPF (Create Display File) command

  The LANGID and SRTSEQ parameters are supported. The values of the RANGE, VALUES, and COMP keywords are validated when the display file is compiled.
- High-level languages

  Using ILE COBOL and ILE RPG IV languages, you can specify SRTSEQ and LANGID values directly on the Create Bound Program (CRTBNDXXX) commands. Original Program Model RPG and COBOL compilers use the Create Program (CRTXXXPGM) commands. With ILE C, you can also specify SRTSEQ and LANGID values when you create a locale. You can then associate the locale with a program.
- iSeries Access

  The transfer function allows a sort sequence table to be specified when performing queries on database files and SQL tables.

Sort sequence tables: A sort sequence table is an object that contains the weight of each single-byte graphic character within a specified coded character set identifier (CCSID). The system-recognized identifier for the sort sequence table object type is *TBL.

Depending on your requirements, you can define a table to have either a unique weight for each graphic character or shared weights for some graphic characters. If you define a table that contains unique weights for each character within the character set, your table is known as a unique-weight table. If you define a table that contains some graphic characters that share the same weight, your table is known as a shared-weight table. For example, if you want to sort the graphic character capital letter A and the graphic character small letter a together, you would define a shared-weight table. If you want to sort these graphic characters separately, you would define a unique-weight table.

A set of sort sequence tables is shipped with the servers. This set of tables defines both unique- and shared-weight sort sequences for all SBCS languages.

Sort sequence table implementation notes

Sort sequence support does not take into consideration the following:
- Special cases of single characters that should be handled as multiple characters (such as the German character s sharp).
- Sequences of characters that should be treated as a single character (such as the Danish aa, Hungarian ly, Serbian lj, Spanish ll).
- Nonalphanumeric characters that should be ignored because they are embedded in alphanumeric strings (such as the hyphen in co-op).
- Prefixes that should be ignored (such as *Van der* in the name *Van der Pool*).
- Program-described files.
- DBCS code pages.

If a sort sequence table has a weight other than hexadecimal 40 assigned to the blank character, unpredictable results can occur when strings of unequal lengths are compared.

Sort sequence tables shipped with the system

You can use the WRKTBL command to view the contents of the sort sequence tables that are shipped with OS/400. The tables are located in the QSYS library.

When looking at these tables, consider the following:

- Several tables shipped with the system represent a single sort sequence, each encoded with a different coded character set identifier (CCSID) value. Not all of the characters in a given sort sequence exist in every CCSID in which the sort sequence is encoded.
- Use the language identifier (LANGID) parameter and the sort sequence (SRTSEQ) parameter to access the unique-weight tables (*LANGIDUNQ) or the shared-weight tables (*LANGIDSHR).
- When using the sort sequence, the relative weights shown in these tables differ from the actual weights in the sort sequence table on the system. The relative weights shown in these tables are examples only.
- The relative unique weight of a character is shown by the order of the characters in the sort sequence table. The relative unique weight is determined by assigning a weight of 1 to the first character in the sort sequence table and incrementing by 1 for each of the following characters until the end of the table is reached.
- GCGID is the graphic character global identifier.

For example, the Arabic sort sequence table shows the relative sort sequence weights for characters that are sorted using the Arabic sort sequence table.

How to build sort sequence tables

To create a user-defined sort sequence table, copy an existing sort sequence table using the Create Table (CRTTBL) command, and then modifying the copy of the table. Table functions allow you to do the following:

- Use a definition stored in a source member.
- Create a table based on another sort sequence table using an interactive interface.

You can create a sort sequence table (MYTEST) from a copy of an existing table using the following CRTTBL command:

```
CRTTBL TBL(MYTEST) SRCFILE(*PROMPT) TBLTYPE(*SRTSEQ)
BASESRTSEQ(QSYS/QLA10025S) CCSID(037)
```

This command displays a sort sequence table that you can modify. Your table is created from a function key on this display. Your resulting table has a coded character set identifier (CCSID) value of 00037. The table is named MYTEST and is stored in the current library.

The following table shows one way in which the resulting characters may be shown on the first display of the MYTEST sort sequence table. The actual panel shows characters instead of text descriptions. For example, the character shown for sequence 0100 would be a question mark (?), and the character shown for sequence 0070 would be a colon (:).

Note: The characters that you actually see on the first display of the MYTEST sort sequence table may vary, depending on the device that you use.

Sequence	Character
0010	Equal sign
0020	Overline
0030	(SHY)
0040	Hyphen
0050	Comma
0060	Semi-colon
0070	Colon
0080	Exclamation mark
0090	Inverted exclamation mark

Sequence	Character
0100	Question mark
0110	Inverted question mark
0120	Slash
0130	Period
0140	Acute accent mark
0150	Grave accent mark
0160	Caret
0170	Right square bracket
0180	Tilde
0190	Small multiply dot
0200	Comma

You can make changes to the tables to move characters in each code page to the preferred position for the national language sort sequence table. The ordering is done by increments of 10. Therefore, the first value is 10, then 20, and so on. If some characters have a shared weight, these groups of characters have the same sequenced weight.

CCSID support for messages

You can use CCSID support for handling messages and message catalogs on OS/400. You can send messages tagged with one CCSID to users with a different CCSID. You can use CCSID support to handle messages by using commands and application programming interfaces.

Note: You do not need a multinational character set (MNCS) when using CCSIDs for handling messages.

For example, if you do not set CCSID support on, the following message, encoded in CCSID 00037:

```
Joe, I need to see you right away!
```

appears to a user with CCSID 00500 as

```
Joe, I need to see you right away]
```

Instead of seeing an exclamation mark (!), Joe sees a right square bracket (]). If you set CCSID support on, the text in a message encoded in CCSID 00037 is converted to CCSID 00500. Both the person sending the message and the person receiving the message see identical text.

CCSID support helps preserve data integrity in messages. As you read through this information, you will see other advantages to using CCSID support for messages.

Object-level CCSIDs

- Message files
  - Message-level support
  - Message description-level support
- Message queues
- Job message queues
- System reply lists
- History log

The following topics provide detailed information about message support:

- Setting up CCSID support for message handling

- CCSID support for message catalogs
- Converted character replacement data type field
- Retrieve messages
- Receive messages
- Common questions about CCSID support for message handling

For more information

See Message catalogs for a description of general OS/400 globalization support for messages.

The following message handling commands support CCSIDs:
- CRTMSGF (Create Message File)
- CRTMSGQ (Create Message Queue)
- CHGMSGQ (Change Message Queue)
- ADDRPYLE (Add Reply List Entry)
- CHGRPYLE (Change Reply List Entry)
- CHGMSGD (Change Message Description)
- RTVMSG (Retrieve Message)
- RCVMSG (Receive Message)
- SNDBRKMSG (Send Break Message)
- SNDMSG (Send Message)
- SNDPGMMSG (Send Program Message)
- SNDRPY (Send Reply)
- SNDUSRMSG (Send user Message)

Handle messages with a specific object-level CCSID: The following objects support CCSIDs:
- Message files
- Message queues
- Job message queues
- System reply lists
- History log

Each of these objects has an object-level CCSID. The object-level CCSID is the CCSID in which all the messages in that object are encoded.

See the following topics for details about object-level CCSIDs:
- Object-level coded character set identifiers 65535 and 65534
- Using a specific object-level CCSID for handling messages

Object-level coded character set identifier 65535: CCSID 65535 is the default object-level CCSID for message files and message queues. If an object has a CCSID of 65535, no conversions occur when adding messages to that object or when receiving messages from that object. Use CCSID 65535 if you do not want CCSID processing to occur.

CCSID 65535 is also known as *HEX.

Object-level coded character set identifier 65534: CCSID 65534 is the default object-level CCSID for job message queues, system reply lists, and the history log. If the CCSID of an object is 65534, each message in the object has its own CCSID. No conversion occurs when a message is added to the object. When a message is received, it is converted based on the CCSID stored with the message.

CCSID 65534 is also known as *MSG or *MSGD.

CCSID 65534 is the preferred setting for object-level CCSIDs. An object-level CCSID of 65534 requires fewer CCSID conversions. Fewer CCSID conversions of text result in better performance and improved data integrity.

Using a specific object-level CCSID for handling messages: If the CCSID of an object is any value other than 65535 or 65534, all messages in that object are considered encoded in that CCSID. The object-level CCSID overrides the CCSID stored with the messages. Use this type of object-level CCSID if both of the following are true:

- You expect the object to be sent messages or have message descriptions added in a CCSID different from the CCSID in which you would receive the messages or retrieve the message descriptions.
- You intend to receive the same message or retrieve the same message description many times.

If these conditions are true, set the object-level CCSID to the CCSID in which you will receive or retrieve the messages. When the system uses this type of object-level CCSID, the message text or data is converted at the time the message is sent or is added to the object. No conversion occurs when the message is received or retrieved because the text and data are already in the CCSID requested on the receive operation or retrieve operation.

Do not change system-supplied message files to use this type of object-level CCSID. Each system-supplied message description is tagged separately. No one object-level CCSID value can represent all of the message descriptions in the message file. Changing the object-level CCSID of a system-supplied message file to anything other than CCSID 65535 or CCSID 65534 may cause unpredictable results.

Message-level support: When a message is sent to a message queue, you must communicate the CCSID of the replacement data or the impromptu message text to the operating system. Use the CCSID parameter on any of the send message commands or APIs to communicate this CCSID to the operating system.

The default CCSID setting in the send message commands and APIs indicate that the replacement data or impromptu message text is in the CCSID of the job that is running the command or API. You can override the job default CCSID value by specifying a different CCSID value.

If the replacement data or impromptu message text supplied is not in the CCSID specified, incorrect conversion results may occur. See Can I correct the CCSID of a message? if this occurs.

Determining the CCSID of a message file

To determine the CCSID of a message file, type:

```
WRKMSGD MSGF(MYLIB/MYMSGF)
```

where MYLIB is the library in which the message file is stored and MYMSGF is the name of the message file.

Next, press F22 (Display list details).

You can also use the QMHRMFAT (Retrieve Message File Attributes) application program interface (API) to determine the CCSID of a message file.

For job message queues, system reply lists, and the history log, the object-level CCSID is always 65534. You cannot change nor display object-level CCSIDs for job message queues, system reply lists, and the history log.

How the message-level CCSID is set

See the following topics for information on how the message-level CCSID is set:

- Message-level CCSID with a message queue CCSID of 65535 or 65534
- Message-level CCSID with a specific message queue CCSID
- Message-level CCSID when a message queue CCSID conversion error occurs
- Message-level CCSID when a message is a stored message

Message-level CCSID with a message queue CCSID of 65535 or 65534: When a message is sent to the message queue and the CCSID of the message queue is 65535 or 65534, no conversion occurs on the message. The message-level CCSID is set to the CCSID specified.

For example, message queue MYMSGQ has a CCSID of 65534. You enter the following Send Message command:

```
SNDMSG MSG('MSG #1') CCSID(37) TOMSGQ(MYLIB/MYMSGQ)
```

The impromptu message text, MSG #1, is not converted when added to the message queue. The message is tagged with CCSID 00037.

Message-level CCSID with a specific message queue CCSID: When a message is sent to the message queue and the CCSID of the message queue is something other than 65535 or 65534, the replacement data or impromptu message text is converted to the CCSID of the message queue. The message is then tagged with the CCSID of the message queue.

For example, message queue MYMSGQ has a CCSID of 00277. The replacement data for TST0002 is defined as *CCHAR data. You enter the following Send Program Message command:

```
SNDPGMMSG MSGDTA(X'0006D4E2C7407BF2') MSGID(TST0002) MSGF(MYMSGF)
CCSID(37) TOMSGQ(MYLIB/MYMSGQ)
```

The replacement data is converted from CCSID 00037 to CCSID 00277 before it is sent to the message queue. X'0006' is the length required for variable-length fields. X'D4E2C7407BF2' is *MSG #2* on code page 00037. The number sign (#), X'7B' on code page 00037, is converted to a number sign, X'4A' on code page 00277. All other code points do not change during the conversion because they are the same on both code page 00037 and code page 00277.

When the replacement data or impromptu message text of a message is 65535 and it is sent to a message queue with a CCSID other than 65535 or 65534, no conversion occurs. However, the message is tagged with the CCSID of the message queue. Therefore, messages can be tagged with an incorrect CCSID when you send them to a message queue with a CCSID that overrides the message-level CCSID.

For example, message queue MYMSGQ has a CCSID of 00277. You enter the following Send Message command:

```
SNDMSG MSG('MSG #2') TOMSGQ(MYLIB/MYMSGQ) CCSID(*HEX)
```

The impromptu message text *MSG #2* is not converted before it is sent to the message queue. Although the impromptu message text is not converted to CCSID 00277, it is displayed using CCSID 00277. Unless you entered the Send Message command from a device configured to support code page 00277, you lost the integrity of the impromptu message text.

Message-level CCSID when a message queue CCSID conversion error occurs: If a conversion error occurs while sending a message to a message queue, the message is still sent to the message queue.

However, the impromptu text or data of the message is not converted. A diagnostic message is sent and the message is tagged with the message-level CCSID specified on the send command or API, not with the CCSID of the message queue.

You can recover the replacement data or impromptu message text with the proper CCSID setting. First, set the message queue CCSID to 65534. Then use the Receive Message command or API to return the correct message-level CCSID.

Message-level CCSID when a message is a stored message: If a message is a stored message, the message-level CCSID applies only to *CCHAR replacement data. The CCSID of the first- and second-level text of the message is retrieved from the message file.

Replies to stored messages are never converted from one CCSID to another. Only replies to impromptu messages are affected by CCSID processing.

Message description-level support: When a message description is added to a message file, the CCSID of the message text must be communicated to the operating system. You can use the CCSID parameter on the ADDMSGD or the CHGMSGD command to communicate this CCSID to the operating system.

The default settings of these commands indicate that the message text is in the CCSID of the job that is running the command. You can change this value by specifying a different CCSID value. You can also change this value by indicating that no CCSID processing should occur. You indicate that no CCSID processing should occur on the message text by specifying a CCSID value of 65535 (*HEX).

If you set CCSID processing on, system-supplied display files and printer files that display or print message descriptions convert the CCSID of the message file to the CCSID of the job before displaying them or printing them. To print and display the messages correctly, your job CCSID setting must be the same as the code page portion of your device CHRID setting.

All message descriptions that existed in a message file that was created prior to V3R1 are tagged with CCSID 65535 on the first use or handling of that message description.

If the text of a message is not in the CCSID specified, incorrect conversion results may occur. See Can I correct the CCSID of a message description? if this occurs.

How the CCSID of a message description is set

To set the message description-level CCSID, consider the following topics:

- Message file with a CCSID of 65535 or 65534
- Message file with a specific CCSID

How to change the CCSID of a message description

See Changing the CCSID of a message description for details.

Message file with a CCSID of 65535 or 65534: If the CCSID of the message file is 65535 or 65534, no conversion occurs on the message description when it is added to the file. The message description CCSID is set to the CCSID specified on the ADDMSGD or CHGMSGD command.

For example, a message file MYMSGF has a CCSID of 65534. The job that is running is in CCSID 00037. You enter an ADDMSGD command, as follows:

```
ADDMSGD MSG('MSG #1') MSGID(TST0001) MSGF(MYMSGF)
```

The message text, MSG #1, is not converted when added to the message file. The message text is tagged 00037 because the CCSID parameter was not coded on the ADDMSGD command and the default CCSID parameter is *JOB.

Message file with a specific CCSID: If the CCSID of the message file is something other than 65535 or 65534, the first- and second-level text of the message description is converted from the CCSID specified to the CCSID of the message file. It is then tagged with the CCSID of the message file.

For example, message file MYMSGF has a CCSID of 00277. The job that is running is in CCSID 00037. You enter the following command:

```
ADDMSGD MSG('MSG #2') MSGID(TST0002) MSGF(MYMSGF) CCSID(37)
```

Message 'MSG #2' is converted from CCSID 00037 to CCSID 00277 before it is added to the message file. The number sign (#), X'7B' on code page 00037, is converted to the number sign (#), X'4A', on code page 00277. No other code points change during the conversion because they are the same on both code page 00037 and code page 00277.

When the text of a message description is specified as 65535 and it is added to a message file, no conversion occurs. If the CCSID of the message file is not 65535 or 65534, the message text is tagged with the CCSID of the message file.

When the message file CCSID is not 65535 or 65534, the message file CCSID overrides message description CCSIDs. Keep this rule in mind when adding and changing message descriptions to a message file with a CCSID other than 65535 or 65534. Otherwise, a message description can be marked incorrectly.

For example, message file MYMSGF has a CCSID of 00277. You enter the following command:

```
ADDMSGD MSG('MSG #2') MSGID(TST0002) MSGF(MYMSGF) CCSID(*HEX)
```

Message text 'MSG #2' is not converted before it is added to the message file. Because the CCSID of the message file is 00277, the message text is tagged with CCSID 00277.

If the command was run in a job CCSID where the number sign (#) occupies a code point different than the code point for the number sign on code page 00277, the message is displayed incorrectly.

A conversion error may occur while adding or changing a message description in a message file. If a conversion error occurs, the message description is still either added to or changed in the message file. The text of the message description, however, is not converted. A diagnostic message is sent and the message description is tagged with the CCSID specified, not with the CCSID of the message file.

When a conversion error occurs, you can recover the correct CCSID tagging for the message description by setting the message file CCSID to 65534. Then you can retrieve the correct CCSID for the message description using the Retrieve Message (RTVMSG) command or the Retrieve Message (QMHRTVM) API.

The CCSID of a message description applies only to first- and second-level message text.

Change the CCSID of a message description: When you take the option to change a message description from the Work with Message Descriptions display, all current values for the selected message description are retrieved and placed on the prompt display. The first- and second-level text are converted from the CCSID of the message file to the CCSID of the job before they are put on the prompt display.

*JOB is displayed for the CCSID keyword and has two different meanings depending on what you do on the prompt display. If you change any part of the first- or second-level text, *JOB means that the text is converted from the CCSID of the job to the CCSID of the message file when you press the Enter key. If the text is unchanged, *JOB works like *SAME, and none of the following are changed:

- The first-level message text

- The second-level message text
- The CCSID of the message description

Both the first- and second-level text of a message description must be in the same CCSID. If you change the CCSID of one level, the system automatically converts the other level to match.

Example: Changing a message description

The CCSID of message file MYMSGF is 65534. The CCSID of the job that is running WRKMSGD is 00277. The CCSID of the message description is 00037.

Select option 2 to change a message description. The text of the message description is converted from CCSID 00037 to 00277 before being placed on the prompt display.

If only the first-level text is changed, the 00277-tagged text is stored in the message file. The CCSID of the message description is changed to 00277. The 00277-tagged second-level text is also stored in the message file to keep both the first- and second-level text in the same CCSID.

Message queues: If you set CCSID processing on, system-supplied display files and printer files that display or print messages convert the CCSID of the message queue to the CCSID of the job before displaying or printing the messages. To print and display the messages correctly, your job CCSID setting must be the same as the code page portion of your device CHRID setting.

All messages that existed on a message queue that was created in a release prior to V3R1 are assigned CCSID 65535 on the first use of that message.

Determining the CCSID of a message queue

To determine the CCSID of a message queue, type:

```
DSPMSG MSGQ(MYLIB/MYMSGQ) ASTLVL(*BASIC)
```

where MYLIB is the library in which the message queue is stored and MYMSGQ is the name of the message queue.

Then press F22 (Display list details).

You can also use the Retrieve Message Queue Attributes (QMHRMQAT) application program interface (API) to determine the CCSID of a message queue.

For job message queues, system reply lists, and the history log, the object-level CCSID is always 65534. You cannot change nor display object-level CCSIDs for job message queues, system reply lists, and the history log.

Job message queues: The CCSID for all job message queues is 65534. You cannot change or display this value. A job message queue CCSID of 65534 requires fewer CCSID conversions. Fewer CCSID conversions of text result in better performance and improved data integrity.

The CCSID of each message in the job log is used for CCSID processing. No conversion occurs when a message is sent to the job log.

Note: Request messages are always tagged with a CCSID of 65535 and are never converted.

If you set CCSID processing on, system-supplied display files and printer files that display or print job logs convert the CCSID of the messages to the CCSID of the job before displaying or printing the messages.

To print and display the messages correctly, your job CCSID setting must be the same as the code page portion of your device CHRID setting. Status messages that appear on line 24 of a display are converted to the CCSID of the device before they are shown.

For more information about Job message queues and CCSID support, see History log.

System reply list: The system reply list has a CCSID of 65534. You cannot change or display this value. The only part of the system reply list that is affected by CCSID processing is the Compare data field. If the Compare data field references replacement data that is defined as *CCHAR, the data being compared must be in a common CCSID before the comparison is done.

Any reply list entry that has compare data is tagged with the CCSID supplied on the ADDRPYLE or CHGRPYLE commands. When the system reply list is used, the replacement data is converted to the CCSID of the compare data before the comparison is made and before the message is sent to the message queue. This ensures that the data is in a common CCSID before the comparison is done.

Example: System reply list and converted-character compare data

Enter the following Add Reply List Entry command:

```
ADDRPYLE SEQNBR(101) MSGID(TST0010) CMPDTA(X'00017B') RPY(*DFT) +
CCSID(37)
```

X'7B' is the number sign (#) on code page 00037. X'0001' is the length required for variable-length fields. The compare data is not converted when added to the system reply list. It is tagged with CCSID 00037. Message TST0010 has one replacement data field that is defined as *CCHAR with (*VARY 2) for its length. Message queue MYMSGQ has a CCSID of 00278.

Send message TST0010 in a job that has the system reply list turned on using the following Send Program Message command:

```
SNDPGMMSG MSGID(TST0010) MSGF(MYLIB/MYMSGF) MSGTYPE(*INQ) +
TOMSGQ(MYLIB/MYMSGQ) MSGDTA(X'00014A') CCSID(277)
```

The replacement data is converted from CCSID 00277 to CCSID 00037 and then compared with the compare data. The conversion results in replacement data X'00017B'. A match is found and the default reply is sent when this message is added to the message queue.

When the message is added to the message queue, the replacement data is converted from CCSID 00277 to CCSID 00278. The message queue CCSID does not matter when trying to match the compare data. The replacement data is converted to X'000163' when it is sent to the message queue and tagged 00278. X'63' is the code point for the number sign (#) in code page 00278.

History log: The history log is a database file that is tagged with CCSID 65535. You cannot change the CCSID of the history log. No conversions occur when you do database retrievals from the history file.

You can use CCSID processing when working with the history log. The CCSID of the replacement data or impromptu message text is added to the history log record. If the history log record is for a stored message, CCSID processing occurs only for the *CCHAR replacement data in that record.

You can retrieve a message from the history log and convert it into a specific CCSID by doing the following:

1. Obtain the input variables &MSGFL, &MSGF, &MSGID, &MSGDTA, and &MDTACCSID, from the history log record. (See the CL Programming

   PDF for the layout of the history log record.)

2. Enter the following Retrieve Message command:

```
RTVMSG MSGF(&MSGFL/&MSGF); MSGID(&MSGID); MSGDTA(&MSGDTA);  +
MDTACCSID(&MDTACCSID); MSG(&MSG);
```

If you set CCSID processing on, system-supplied display files and printer files that display or print history log records convert the CCSID of the messages to the CCSID of the job before displaying or printing the messages. To print and display the messages correctly, your job CCSID setting must be the same as the code page portion of your device CHRID setting.

Set up CCSID support for message handling: The default setting of the CCSID for creating message queues and message files is 65535. Most message files delivered with the operating system have a CCSID of 65535.

Most message descriptions in system-supplied message files are tagged with a CCSID that corresponds to the national language version with which they are shipped.

Some message descriptions are not assigned a CCSID that corresponds to the national language version. These message descriptions are tagged 65535 and are not converted when used.

Messages sent to a message queue that has a CCSID of 65535 are not converted when placed on the queue. Message descriptions added to a message file that has a CCSID of 65535 are not converted when placed in the file. These messages and message descriptions are tagged with a CCSID associated with their text or data. By tagging them with a CCSID associated with their text or data, they are given the correct CCSID if the object-level CCSID is changed to 65534.

You can set CCSID support on for handling a specific message queue. For example, to set CCSID handling on for message queue MYMSGQ in library MYLIB, type:

```
CHGMSGQ MSGQ(MYLIB/MYMSGQ) CCSID(65534)
```

The Change Message Queue (CHGMSGQ) command also allows you to turn on CCSID support for more than one message queue at a time.

You can set CCSID support on for handling a specific message file. For example, to set CCSID handling on for message file MYMSGF in library MYLIB, type:

```
CHGMSGF MSGF(MYLIB/MYMSGF) CCSID(65534)
```

The Change Message File (CHGMSGF) command also allows you to turn on CCSID support for more than one message file at a time.

CCSID support for message catalogs: The Message catalog CCSID (CLGCCSID) parameter allows you to specify the CCSID for storing data in a message catalog. The Source file CCSID (SRCCCSID) parameter allows you to specify the CCSID of a source file. Data from the source is converted to the CCSID of the message catalog if the CCSIDs for both are not the same. This is also the default action. The source can be in any CCSID that supports conversion to any other CCSID.

The CCSID of the original message catalog is used to update the message catalog. It can be single or mixed and in extended binary-coded decimal interchange code (EBCDIC), American National Standard Code for Information Interchange (ASCII), or UCS-2. If the catalog is a QSYS source file member that does not exist, the CCSID of the existing file is used. The value that is specified on the CLGCCSID parameter is used if the CCSID of the file is 65535.

Converted character replacement data type field: A replacement data type field supports CCSID processing. This replacement data type field is called a convertible character field (*CCHAR). A *CCHAR replacement data type field is a variable-length field. This field may increase or decrease in length when the field is converted.

Example: Add a message description with CCSID support

The following example shows how to add the message description TST0006 to message file MYMSGF. The message description has 2 replacement data type fields. One field is a character field length 10. The other field is a convertible character field with varying length. Use the ADDMSGD command as follows:

```
ADDMSGD MSG('This is *CHAR &1;  This is *CCHAR &2;') MSGID(TST0006) +
MSGF(MYLIB/MYMSGF) FMT((*CHAR 10) (*CCHAR *VARY 2))
```

Retrieve messages: The Retrieve Message (RTVMSG) command and retrieve message (QMHRTVM) application program interface (API) have a CCSID-to-convert-to parameter. This parameter determines which CCSID the first- and second-level text is converted to before the text is returned to the user. The Retrieve Message command and the Retrieve Message API also have a replacement data CCSID parameter. This parameter communicates the CCSID of the replacement data to the system. The replacement data CCSID applies only to the parts of the replacement data that correspond to *CCHAR type data. No other replacement data is converted.

The Retrieve Message command and Retrieve Message API convert the first- and second-level text from the CCSID of the message file to the CCSID on the CCSID-to convert-to parameter. Any replacement data that is *CCHAR data is converted from the replacement data CCSID to the CCSID-to-convert-to CCSID before being substituted into the correct replacement variables. The default for both parameters is *JOB, which means that the CCSID of the job is used.

Retrieve Message command CCSID return fields

Three CCSID return fields are supported by the Retrieve Message (RTVMSG) command:

- TXTCCSID
- TXTCCSTA
- MDTACCSTA

Example 1: Retrieving a message with CCSID support

Message file MYMSGF has a CCSID of 65534. The CCSID of the message description is used to determine the CCSID from which to convert the message text. The CCSID of the message description (TST0003) is 00037. The first-level text is:

```
'MSG #3 is &1;'
```

&1 is defined as a *CCHAR variable field with a length of (*VARY 2). Enter the following RTVMSG (Retrieve Message) command:

```
RTVMSG MSGF(MYMSGF) MSGID(TST0003) MSG(&MSG); CCSID(277)  +
MDTACCSID(277) MSGDTA(X'0002D6D2')
```

In the message data, the first 2 bytes are a length field with a value of 2. All *VARY fields begin with a length. The next 2 characters are the actual *CCHAR data with a value of X'D6D2'. X'D6D2' represents the characters *O* and *K* on code page 00277.

The first-level text is converted from CCSID 00037 to CCSID 00277. The replacement data is not converted before it is substituted for &1; because the replacement data CCSID matches the CCSID-to-convert-to parameter. As a result, the text returned in the variable &MSG is:

```
'MSG #3 is OK.'
```

The code point for the number sign (#) is the only change that occurred in the conversion. The number sign was converted from code point X'7B' in code page 00037 to code point X'4A' in code page 00277. All other code points in the text of the message matched in code page 00037 and code page 00277.

Note: If the CCSID of a message file is 65535, no conversion occurs, even though the message description CCSID is 00037. The CCSID of the message file always takes precedence over the message description CCSID.

Example 2: Using return fields and converted character data

Message description TST0005 has the following first-level text:

```
This is *CHAR &1;  This is *CCHAR &2;
```

The message description is defined in message file MYMSGF, which has a CCSID of 65535. &1; is defined as a *CHAR field of length 1. &2; is defined as a *CCHAR field (*VARY 2) in length. The CCSID of the message description does not matter because the CCSID of the message file is not 65534. You enter the following RTVMSG command:

```
RTVMSG MSGF(MYMSGF) MSGID(TST0005) MSG(&MSG); CCSID(260)  +
MDTACCSID(37) MSGDTA(X'5A00015A') TXTCCSID(&TXTCCSID);
```

Note: X'5A' is the exclamation point (!) on code page 00037.

These are the returned values from the RTVMSG command:

- &MSG = 'This is *CHAR. This is *CCHAR !.'

  The EBCDIC value of the *CHAR character is X'5A'. X'5A' appears as an acute accent (

  ´

) on code page 00260. The *CHAR data did not convert because only *CCHAR data supports CCSID processing. The '&1' stayed at X'5A', while '&2' converted to X'4F'. X'4F' is the exclamation point on code page 00260.
- &TXTCCSID = 65535

  The TXTCCSID variable is set to 65535 because no conversion occurred. When no conversion occurs, the CCSID (if it is not 65534) of the message file is returned.

CCSID of the text returned (TXTCCSID) return field: TXTCCSID is the CCSID of the text returned. If a conversion occurs and is successful, this value is always equal to the CCSID-to-convert-to value. If a conversion occurs and is not successful, this is the CCSID of the message file unless the CCSID of the message file is 65534. If the CCSID of the message file is 65534, the CCSID of the message description is returned.

For example, message file MYMSGF has a CCSID of 65534. Your program needs to know the CCSID of message description TST0004. Specify the RTVMSG command as follows:

```
RTVMSG MSGF(MYMSGF) MSGID(TST0004) CCSID(*HEX) TXTCCSID(&TXTCCSID);
```

The CCSID of the message description is returned in the variable &TXTCCSID because you specified *HEX for the CCSID-to-convert-to parameter. *HEX means no conversion is to occur. If no conversion occurs and the message file CCSID is 65534, the message description CCSID is returned.

You can also obtain the message description CCSID from the Work with Message Descriptions (WRKMSGD) display.

1. On the WRKMSGD display, select option 5 to display details.
2. From the Select Message Details to Display menu, select option 5 to display message attributes.
3. Page forward to the CCSID value. The message description CCSID is shown if the CCSID of the message file is 65534. If the CCSID of the message file is not 65534, the CCSID of the message file is shown.

CCSID conversion status indicator (TXTCCSTA) return field: TXTCCSTA is the text CCSID conversion status indicator. Return codes help you determine what happened when the system converted your

message text to the CCSID-to-convert-to parameter. Positive return code numbers indicate that your conversion was successful. A successful return code does not always indicate that a conversion occurred. Negative return code numbers indicate that a conversion error occurred.

The following list shows the available return codes:

0 No conversion was necessary. The CCSID of the text matched the CCSID that you wanted the text converted to.

1 No conversion occurred. Either the text was 65535 or the CCSID that you wanted the text converted to was 65535.

2 No conversion occurred. You did not ask for any text to be returned.

3 The text was converted to the CCSID specified. The conversion operation used the linguistic conversion tables.

4 A conversion error occurred when the conversion operation used the linguistic conversion tables. The conversion operation then used a default conversion table. The default conversion completed without error.

-1 An error occurred on both the linguistic and default conversions. The text was not converted.

Replacement data CCSID conversion status indicator (MDTACCSTA) return field: MDTACCSTA is the replacement data CCSID conversion status indicator. Return codes help you determine what happened when the system converted your replacement data to the CCSID-to-convert-to parameter.

Positive return code numbers indicate that your conversion was successful. A successful return code does not always indicate that a conversion occurred. Negative return code numbers indicate that a conversion error occurred. These return codes are similar to the TXTCCSTA return codes. The return codes apply to the conversion that takes place on any *CCHAR replacement data being converted from the replacement data CCSID to the CCSID-to-convert-to value.

The following list shows the available return codes:

0 No conversion was necessary. The CCSID of the replacement data matched the CCSID that you wanted the text converted to.

1 No conversion occurred. Either the replacement data was 65535 or the CCSID that you wanted the replacement data converted to was 65535.

2 No conversion occurred. Either you did not ask for any replacement data to be returned or no *CCHAR replacement data fields were defined for the message description being retrieved.

3 The replacement data was converted to the CCSID specified. The conversion operation used the linguistic conversion tables.

4 A conversion error occurred when the conversion operation used the linguistic conversion tables. The conversion operation then used a default conversion table. The default conversion completed without error.

-1 An error occurred on both the linguistic and default conversions. The replacement data was not converted.

Receive messages: The Receive Message (RCVMSG) command, the Receive Nonprogram Message (QMHRCVM) API, and the Receive Program Message (QMHRCVPM) API have a CCSID-to-convert-to parameter. This parameter determines which CCSID the text or data is converted to before it is returned to the user.

The Receive Message command and APIs convert the text or data from the CCSID of the message queue or message file to the CCSID supplied on the CCSID-to-convert-to parameter. When replacement data is returned, only the *CCHAR data is converted from the CCSID of the message queue to the CCSID-to-convert-to value.

If the CCSID of the message file or message queue is 65534, the text or data is converted from the CCSID of the message description or message to the CCSID supplied on the CCSID-to-convert-to parameter.

The default for the CCSID-to-convert-to parameter is *JOB, which means that the CCSID of the job performing the receive operation is used.

Receive Message command CCSID return fields

Two CCSID return fields are supported by the Receive Message (RCVMSG) command:

- TXTCCSID
- DTACCSID

Receive Message API CCSID return fields

The Receive Message (QMHRCVM) API and the Receive Program Message (QMHRCVPM) API support the return fields defined in TXTCCSID return field for receive message command and DTACCSID return field for receive message command. The Receive Message API and the Receive Program Message API also support two additional return fields.

Example 1: Using the CCSID return fields

Message description TST0005 has the following first-level text:

```
This is &CHAR &1;  This is *CCHAR &2;
```

'&1' is defined as a *CHAR field of length 1. '&2' is defined as a *CCHAR field (*VARY 2) in length.

Message file MYMSGF has a CCSID of 65534. TST0005 is defined in message file MYMSGF. The message description CCSID is 65535. The CCSID of message queue MYMSGQ is 65534.

You enter the following Send Program Message command:

```
SNDPGMMSG MSGF(MYMSGF) MSGID(TST0005) CCSID(37) TOMSGQ(MYLIB/MYMSGQ) +
MSGDTA(X'7B00017B')
```

The message is not converted when it is sent to message queue MYMSGQ because the message queue CCSID is 65534. The message is tagged with CCSID 00037.

You enter the following Receive Message command to receive the message just sent:

```
RCVMSG MSGQ(MYLIB/MYMSGQ) MSG(&MSG); DTACCSID(&DTACCSID); +
CCSID(277) MSGDTA(&MSGDTA); TXTCCSID(&TXTCCSID);
```

Note: X'7B' is the number sign (#) on code page 00037.

Because the message description is tagged 65535, no conversion is performed when retrieving the message text of TST0005. The replacement data is tagged 00037. The *CCHAR part of the message data is converted from CCSID 00037 to CCSID 00277 before being inserted for &2; *CHAR data is never converted.

The following table shows the returned values after the Receive Message command runs:

Value	Description
&MSG =	`This is &CHAR .  This is *CCHAR #.` The *CHAR data was not converted when substituted for &1; The *CHAR data remains X'7B'. X'7B' is the code point on code page 00277 for A ligature (Æ). The *CCHAR data was converted to X'4A' before it was substituted for &2; X'4A' is the code point on code page 00277 for the number sign (#).
&TXTCCSID = 65535	The &TXTCCSID variable was set to 65535 because no conversion occurred. When no conversion occurs, the CCSID of the message description is returned if the CCSID of the message file is 65534.
&DTACCSID = 00277	The &DTACCSID variable was set to 00277 because a conversion occurred.

Example 2: Receiving a message with CCSID support

Message file MYMSGF has a CCSID of 00037. Message queue MYMSGQ has a CCSID of 65534. The message being received has a message-level CCSID of 00277. CCSID 65534 uses the message-level CCSID when determining the CCSID the replacement data is to be converted from.

The message being received is a stored message. The stored message has *CCHAR replacement data. The CCSID of the job is 00278. You enter the following Receive Message command:

```
RCVMSG MSGQ(MYMSGQ) MSG(&MSG); MSGDTA(&MSGDTA);
```

The first-level text of the stored message that you receive is converted from CCSID 00037 to CCSID 00278. The replacement data of the message that you receive is converted from CCSID 00277 to CCSID 00278. Then the replacement data is substituted into the first-level text and returned in &MSG.

Both the first-level text and the replacement data of the message that you received are converted to the CCSID of the job because the CCSID of the job is the default for the CCSID-to-convert-to parameter.

Two different conversions must occur because only the replacement data is stored in the message queue for stored messages. The text of a stored message must be retrieved from the message file. If the message contained other replacement data type fields that were not defined as *CCHAR, the non-*CCHAR data is not converted before being returned.

Note: If the CCSID of the message queue is 00278, no conversion occurs on the replacement data before the message is returned, even though the message CCSID is 00277. Remember that the message queue CCSID takes precedence over the message-level CCSID.

CCSID of the message text returned (TXTCCSID) return field: TXTCCSID is the CCSID of the message text returned. If a conversion occurs and the conversion is successful, this value is always the same as the CCSID-to-convert-to value.

For impromptu text, if the conversion is not successful, TXTCCSID is the CCSID of the message queue, unless the message queue is 65534. If the message queue is 65534, TXTCCSID is the message-level CCSID of the impromptu text.

For a stored message, if the conversion is not successful, TXTCCSID is the CCSID of the message file that contains the stored message, unless the message file is 65534. If the CCSID of the message file is 65534, TXTCCSID is the CCSID of the message description for the stored message.

CCSID of the replacement data returned (DTACCSID) return field: DTACCSID is the CCSID of the replacement data returned. DTACCSID applies only to those parts of the replacement data defined as *CCHAR. If the message being received is an impromptu message, a value of 0 is returned. If a conversion occurs and the conversion is successful, this value is the same as the CCSID-to-convert-to value.

If the conversion is not successful, the DTACCSID returned is the CCSID of the message queue, unless the CCSID of the message queue is 65534. If the CCSID of the message queue is 65534, the DTACCSID returned is the CCSID of the message.

For example, a stored message TST0004 from message file MYMSGF is sent to message queue YOURMSGQ with replacement data. TST0004 is defined with *CCHAR replacement data. Message file MYMSGF is 65534. Message queue YOURMSGQ has a CCSID of 00037.

Your program needs to know the CCSID of the message description and the replacement data sent to message queue YOURMSGQ. You enter the following Receive Message command:

```
RCVMSG MSGQ(YOURMSQ) CCSID(*HEX) TXTCCSID(&TXTCCSID); DTACCSID(&DTACCSID);
```

The message description CCSID is returned in the variable &TXTCCSID. The message description CCSID is returned because you specified *HEX for the CCSID-to-convert-to parameter. *HEX means that no conversion is to occur. If no conversion occurs and the message file CCSID tag is 65534, the CCSID of the message description is returned.

The CCSID of message queue YOURMSGQ (00037) is returned in the variable &DTACCSID. The message queue CCSID is returned because it is not 65534.

You can also obtain the message-level CCSID using the Display Messages (DSPMSG) display.

1. From the Display Messages display, press Help to display the Additional Message Information display.
2. Then press F9 (Display Message Details).

   This displays the message-level CCSID when the CCSID of the message queue that this message is on is 65534. Otherwise, the CCSID of the message queue is displayed.

Common questions about CCSID support for handling messages: Following are some common questions asked about CCSID support for handling messages.

- When is the job default CCSID used for handling messages?
- How can I determine if a message description is defined with *CCHAR?
- Can the length of *CCHAR replacement data change?
- Can I correct the CCSID of a message queue?
- Can I correct the CCSID of a message file?
- Can I correct the CCSID of a message?
- Can I correct the CCSID of a message description?

When is the job default CCSID used for handling messages?: A job default CCSID is always a CCSID with an encoding scheme of 1100 (single-byte EBCDIC) or 1301 (mixed-byte EBCDIC). The job default CCSID is used whenever both of the following are true:

- A conversion occurs from a CCSID with an encoding scheme other than 1100 or 1301 to a job CCSID.
- The job CCSID is 65535.

For example, ASCII data must be converted to a CCSID associated with the data when asked to convert to the CCSID of a job. The job default CCSID is used because it is never CCSID 65535.

*How can I determine if a message description is defined with *CCHAR?:* You can use the Work with Message Description (WRKMSGD) command to determine if a message description is defined with

*CCHAR data. You can also use the Retrieve Message (QMHRTVM) API to return the replacement data format fields. For more information, see the System API topic.

*Can the length of *CCHAR replacement data change?:* The length of *CCHAR replacement data can change. This is why *CCHAR replacement data is required to be a variable-length field. The length of the field will grow when converting from an SBCS CCSID to the UCS-2 Level-1 CCSID. The length of the field will shrink when converting from the UCS-2 Level-1 CCSID to an SBCS CCSID.

For example, you define message description TST0011 as 'Printer &1; has error &2;' in message file MYMSGF that has a CCSID of 65535. '&1' is defined as *CCHAR data (*VARY 2) in length. This is the name of the printer. *&2*; is defined as a *CHAR data with a length of 1. This is an error code. Enter the following Send Program Message command to send this message to message queue MYMSGQ:

```
SNDPGMMSG MSGID(TST0011) MSGF(MYLIB/MYMSGF) TOMSGQ(MYLIB/MYMSGQ) +
MSGDTA(X'000400500030F1') CCSID(61952)
```

X'0004' is the length of the variable *CCHAR data. X'00500030' represents the characters P0 in CCSID 61952. If message queue MYMSGQ has a CCSID of 00037, the replacement data is converted to X'0002D7F0F1' before it is sent to the message queue. If message queue MYMSGQ has a CCSID of 65535, the data is not converted when it is sent to the message queue.

Your application programs cannot hard-code the position of the return code in this example. When message queue MYMSGQ has a CCSID of 00037, the return code is 5 bytes into the message text. When message queue MYMSGQ has a CCSID of 65535, the return code is 7 bytes into the message text.

Can I correct the CCSID of a message queue?: You may have a message queue that has a CCSID that does not match the CCSID of the messages on it. This usually results from sending messages with a message-level CCSID of 65535 to a message queue with a CCSID that is not 65534 or 65535.

If all of the messages on a message queue have the same message-level CCSID and you know the message-level CCSID, you can enter the following command:

```
CHGMSGQ MSGQ(MYMSGQ) CCSID(nnnnn)
```

In this example, MYMSGQ is the name of the message queue and nnnnn is the message-level CCSID.

If you do not know the CCSID of all the messages on the queue or if the messages on the queue have different CCSIDs, the message queue should have a CCSID of 65535 or 65534. You can change the message queue CCSID to 65535. Or, you can do the following:

1. Delete all of the messages.
2. Change the CCSID of the message queue to 65534.
3. Send all of the messages again.

Can I correct the CCSID of a message file?: You may have a message file that has a CCSID that does not match the CCSID of the message descriptions in it. This usually results from adding message descriptions with a message-level CCSID of 65535 to a message file with a CCSID that is not 65534 or 65535.

If all of the message descriptions in a message file have the same message-level CCSID, and you know the message-level CCSID, you can enter the following command:

```
CHGMSGF MSGF(MYMSGF) CCSID(nnnnn)
```

In this example, MYMSGF is the name of the message file and nnnnn is the message-level CCSID.

If you do not know the CCSID of all the message descriptions in the file or if the message descriptions in the file have different CCSIDs, the message file should have a CCSID of 65535 or 65534. You can handle this situation in either of the following ways:

- Change the CCSID of the message file to 65535.
- Follow these steps:
  1. Change the CCSID of the message file to 65534.
  2. Change the message-level CCSID of each message description to the correct value. See Can I correct the CCSID of a message description? for information on how to correct the CCSID of a message description.

Can I correct the CCSID of a message?: You cannot correct the message-level CCSID of a message. You can change the message queue CCSID to match the message-level CCSID. You can also delete the message and send it again with the correct message-level CCSID.

Can I correct the CCSID of a message description?: You can use the Change Message Description (CHGMSGD) command to change the CCSID of a message description. If you do not change the first- or second-level text at the same time that you change the message description CCSID, the text remains unchanged. Only the CCSID changes.

For example, you can enter the following Change Message Description command to correct the CCSID of a message description without changing any of the first- or second-level message text:

```
CHGMSGD MSGF(MYLIB/MYMSGQ) MSGID(TST0001) CCSID(37)
```

Work with bidirectional data

Arabic and Hebrew languages use an alphabet written and read from right to left. Numerics and Latin text imbedded in the right-to-left text are written and read from left to right; therefore, these languages are called bidirectional languages.

Because bidirectional languages are written and read from right to left, you should avoid using the terms left and right. For example, *right margin* in Hebrew or Arabic documents would be the beginning of the line and not the end. Use the words *start* and *end* in place of the words *right* and *left*.

Hebrew and Arabic have no case-sensitive characters. To avoid the incorrect presentation of characters, no case-sensitive checking or substitution should be performed. In addition, the Arabic language does not use abbreviations, therefore, you should use only complete words.

The following links provide additional information on Bidirectional application design:

- Bidirectional application support
- Checklist: Bidirectional support

See Code globalized applications that use bidirectional data: guidelines for information about how you can design your applications to accommodate bidirectional data.

Bidirectional application support

OS/400 provides bidirectional application support in the following ways:

- Workstation
- Display file
- UIM

Workstation support

Workstations that have the ability to display Arabic and Hebrew character sets also have the ability of right-to-left cursor movement. Right-to-left cursor movement on input fields can be achieved in one of the following ways:

- Pressing a special function key available on Hebrew and Arabic keyboards called the reverse key. This is a toggle function that moves the cursor to the other side of the field, allows for cursor movement in the opposite direction, and also changes the language layer from Latin to Hebrew or Arabic and back again.
- Using the DDS cursor control codes for display files. When the CHECK keyword is used with a cursor-controlled code, it specifies that the cursor is to move from right to left. The valid cursor control codes are:
  - CHECK (RL): Moves the cursor from right to left in specified nonnumeric input fields or in all nonnumeric input fields on the display.
  - CHECK (RLTB): Moves the cursor from right to left between fields.

  When using these parameters, remember the following:
  - Modulus check digit verification is supported, but the check digit is the byte to the extreme right of the field.
  - A field for which right-to-left cursor movement is specified can occupy more than one line on the display. However, the cursor still moves from the top of the display to the bottom.
  - You cannot use right-to-left cursor movement with user-defined data streams.

  Note: If no cursor positioning is specified in the display file or by the program, the cursor is placed in the input-capable field to the extreme left of the top line.

  See the DDS Reference: Concepts topic for more information.

Display file support

The server does not check to make sure that all display files opened to the display station are capable of right-to-left cursor movement. Therefore, it is the responsibility of application programmers to ensure that the proper display files are used.

User interface manager support

The user interface manager gives the following bidirectional support for creating online information and panels:

- BIDI= NONE | RTL | LTR

  This attribute controls the directional orientation of the panels in the panel group.

  RTL indicates that the panel in the panel group is bidirectional and should be displayed with a right-to-left orientation.

  LTR indicates that the panel in the panel group is bidirectional and should be displayed with a left-to-right orientation.
- :RT and :ERT

  Reverse-direction-text tags indicate that the enclosed text has an orientation that is opposite to the orientation of the panel group.

  For a list of UIM tags, see the Application Display Programming

  PDF.

Checklist: Bidirectional support guidelines

When creating an application with bidirectional support, you must follow some guidelines. Some of these guidelines are listed in the following table:

Complies	Not applicable	Rule
		Software design must allow for bidirectional data to be passed to applications in the same order that a speaker of the language would spell it out.
		The product design must allow for the implementation of the correct handling of bidirectional keyboard and presentation functions.
		Designing of a function that implies logical movement of cursor or characters must permit mirroring of that function.
		Keys or operations labeled with directional icons or symbols must perform according to the icon or symbol.
		Keyboard nomenclature for mirrored functions must be independent of the direction of data or text entry.
		Display functions must not assume a left-to-right orientation.
		Field attributes must contain room for directional information.
		Indicator location must be reserved for the current direction of the cursor (direction of input).
		The design must allow for independent handling of graphic and text orientation.
		Provision must be made to allow shape determination to be performed.
		The deshaping must be definable.
		Provision must be made to allow the selection of the appropriate presentation shape for the numerals.
		Characters must be allowed to touch each other on printers and displays.
		Indicator locations should be reserved for screen and field orientation, current level of nesting, status of push (nesting mechanism), and status of symmetric swapping.
		The design should provide for a method to indicate to the user the nesting structure of a string.
		A system-wide method of deshaping Arabic characters or character strings should be provided.
		An indicator location should be provided for the status of shape determination.
		A method should be provided so that proportional spacing can be provided.
		A method should be provided to allow alignment of the baseline of Arabic and Latin characters (including Hindi and Arabic shapes for numerals).

Work with DBCS data

The following topics describe how you handle DBCS data in applications that use DBCS-capable device files:

- Checklist: DBCS application design
- Develop applications that process DBCS data
- DBCS code schemes
- DBCS font tables
- DBCS font files

- DBCS sort tables
- DBCS field definition

A DBCS file is a file that contains double-byte data or is used to process double-byte data. Other files are called alphanumeric files. You can view DBCS files on display, printer, tape, diskette, and ICF devices.

You use data description specifications (DDS) to describe DBCS-capable device files. For information about using DDS, see the DDS Reference: Concepts topic.

You should indicate that a file is DBCS in one or more of the following situations:
- The file receives input, or displays or prints output, which has double-byte characters.
- The file contains double-byte literals.
- The file has double-byte literals in the DDS that are used in the file at processing time (such as constant fields and error messages).
- The DDS of the file includes DBCS keywords.
- The file stores double-byte data (database files).

DBCS strings in a mixed data stream

Usually, both single-byte characters and double-byte characters are used in a DBCS environment. For example, an accounting firm in Japan uses both English and Japanese for the spreadsheet. If both English and Japanese are being encoded as mixed SBCS and DBCS, the product must be able to understand a mixed character set that contains both single-byte coded characters and double-byte coded characters.

In IBM systems that use EBCDIC, a DBCS string is bracketed in a mixed data stream by a shift-out (SO) control character and a shift-in (SI) control character.

The following example shows the coding for a mixed string:

```
sss  (SO)  D1D2D  (SI)  ssss
```

The following example shows the coding for a mixed hexadecimal string:

```
818283  0E     41424143  0F  818283
```

Supported code ranges

OS/400 supports Japanese, Korean, Simplified Chinese, and Traditional Chinese character-set code ranges.

Using the iSeries Access family of products, the servers also provide support for these non-IBM personal computer DBCS code pages:
- Republic of Korea National Standard graphic character set (KS)
- Taiwan Industry Standard graphic character set (Big5)
- The People's Republic of China National Standard graphic character set (GB)

Checklist: DBCS application design

When creating an application with double-byte coded character set support, you must follow some guidelines. A complete list of these guidelines, as well as a full description of each guideline, is included in *Volume 1 Designing Enabled Products, Rules and Guidelines* (SE09-8001). For your convenience, a subset of these guidelines is provided in the following table.

Complies	Not applicable	Rule
		Double-byte coded character set code points in the graphic character range must be used only for graphic characters and must not be used for control purposes.
		Single-byte meaning must not be drawn from either byte of double-byte coded data.
		Double-byte coded character set character generators must be capable of producing user-accessible graphic characters.
		The ability to switch between single-byte coded character set and double-byte coded character set and the coexistence of single-byte coded character set and double-byte coded character set in the same session must be possible.
		User-interface text modules for double-byte coded character set systems must be loaded separately from the running code.

Develop applications that process DBCS data

Design your application programs for processing double-byte data in the same way you design application programs for processing alphanumeric data, with the following additional considerations:

- Make sure that the double-byte data is always processed in a double-byte unit and does not split a double-byte character.
- Identify double-byte data used in the database files.
- Design display and printer formats that can be used with double-byte data.
- If needed, provide DBCS conversion as a means of entering double-byte data for interactive applications. Use the DDS keyword for DBCS conversion (IGCCNV) to specify DBCS conversion in display files. Because DBCS workstations provide a variety of double-byte data entry methods, you are not required to use the OS/400 DBCS conversion function to enter double-byte data.
- Create double-byte messages to be used by the program.
- Specify extended character processing so that the system prints and displays all double-byte data.
- Determine whether additional double-byte characters need to be defined. User-defined characters can be defined and maintained using the character generator utility (CGU). Information on CGU can be found in the *ADTS/400: Character Generator Utility*, SC09-1769-00 book.

The following topics provide more detailed information about how you can use DBCS data in your applications:

- Use of double-byte data
- DBCS coding considerations
- Process double-byte characters
- Display support
- Make DBCS-capable printer files
- Copy spooled and nonspooled DBCS files
- Change alphanumeric to DBCS programs
- DBCS text and CL commands
- DBCS conversion
- SQL and DBCS

Use of double-byte data: You can use double-byte data in the following ways:

- As data in files:
  - Data in database files.
  - Data entered in input-capable and data displayed in output-capable fields of display files.

  - Data printed in output-capable fields in printer files.
  - Data used as literals in display files and printer files.
- As the text of messages.
- As the text of object descriptions.
- As literals and constants, and as data to be processed by high-level language programs.

Double-byte data can be displayed only at DBCS displays and printed only on DBCS printers. Double-byte data can be written onto diskette, tape, disk, and optical storage.

Where you cannot use double-byte data:

You cannot use double-byte data in the following ways:
- As OS/400 object names.
- As command names or variable names in control language (CL) and other high-level languages.
- As displayed or printed output on alphanumeric workstations.

Double-byte character size:

When displayed or printed, double-byte characters usually are twice as wide as single-byte characters.

Consider the width of double-byte characters when you calculate the length of a double-byte data field because field lengths are usually identified as the number of single-byte character positions used. For more information on calculating the length of fields containing double-byte data, see the DDS Reference: Concepts topic.

DBCS coding considerations: If you plan to have your application used in a DBCS environment, you should ensure that it is DBCS-enabled. Following are some suggestions to consider when developing your general product design.
- Reserve more expansion space for DBCS textual data translation than you reserve for SBCS textual data translation. (It is possible, however, that the number of bytes used may be reduced when a SBCS sentence is being translated into DBCS.)
- Ensure programs can understand shift-out and shift-in delimiters. Otherwise, EBCDIC mixed-byte character strings cannot be handled.
- Do not enable short responses for DBCS. For short responses, it is difficult to shift in and out of DBCS. The `yes` and `no` are examples of short responses.
- Remember to use the graphic data type G where appropriate.
- Remember that the 5494 remote controller supports the graphic data type.
- Be careful when converting mixed data between DBCS-host code and DBCS-PC code, because the transition may change the data length. Losing and gaining SO and SI character pairs can upset field-length calculations.
- Make sure the double-byte data is always processed in a double-byte unit. Do not split a double-byte character.
- Design the display as well as the print format to avoid the problem of truncation of a double-byte character into two single-byte units.

See the following for additional DBCS coding considerations:
- Creating physical files
- Target physical files
- Using CCSIDs
- Using DDS keywords
- DBCS file data types

- The Katakana code page (00290)
- UCS-2 level-1 DBCS display support

Creating physical files: When creating a physical file, display file, and printer file for a DBCS environment, consider the IGCDTA parameter present in the following commands:

- Create Physical File (CRTPF) command

  If DBCS fields are described in DDS, the system treats the file as a DBCS file. Otherwise, specify *YES for the parameter of the CRTPF command so that the file can contain double-byte character set data. However, the system ignores the IGCDTA parameter value when a value for the RCDLEN parameter is not specified.
- Create Display File (CRTDSPF) and Create Printer File (CRTPRTF) commands

  Specify *YES for the parameter when using the CRTDSPF or CRTPRTF commands to create the externally described files. Then DBCS attributes, in addition to those defined in the DDS, can be specified.

Target physical files: When the CPYSPLF, DSPSPLF, or WRKSPLF commands with OUTPUT(*OUTFILE) option are used under the DBCS version of the OS/400 program, the target physical file must be DBCS-enabled.

Note: The primary language of the system must support the double-byte character set to allow DBCS-enabled applications.

Use the QIGC system value to check if a DBCS version of the system is installed. Because it is set by the system, it cannot be changed. This system value can be referred to in an application program. QIGC can be:

- 0 (DBCS version is not installed)
- 1 (DBCS version is installed)

A DBCS system allows for concurrent use of SBCS and DBCS data. When the QIGC system value is 1, you should not assume all jobs are DBCS.

Using CCSIDs: Use DBCS CCSIDs for DBCS languages. When designing an application to be used in the DBCS environment, consider the following CCSID information:

- If the QIGC system value is set on, system value QCCSID must have the value of a mixed CCSID.
- If the DBCS and SBCS language users are sharing the same system, they may want to store their data in different databases. It is possible to create DBCS-capable and SBCS-capable physical files in the same system. The CCSID parameter on the CRTPF command or the CCSID keyword on the physical file DDS definition can be used to specify the CCSID value that the data is stored in.
- If a CCSID was not explicitly assigned through DDS at file creation time, the database physical file character J (DBCS-only), E (DBCS-either), O (DBCS-open) or G (DBCS-graphic) fields are implicitly assigned a CCSID value.

Using DDS keywords: Consider the following DDS keywords so that you can specify alternative ways to enter data through display files, change input- and output-capable alphanumeric data fields to DBCS data fields, or to specify the special features of the DBCS printer output:

- CHRSIZ (Character Size)

  This printer file keyword can expand the printer characters to twice the normal size (width and height). This keyword can is valid only for IPDS printers and for printer files with a device type of *IPDS or *AFPDS specified.
- CONCAT (Concatenate)

  This keyword can be used only on logical files. This keyword does not support concatenation of a character field and a data type O field.
- DFLIN (Define Line)

The printer file keyword draws horizontal and vertical lines.

- IGCALTTYP (DBCS Alternative Data Type)

  This display and printer keyword is used to change input- and output-capable character fields to DBCS fields with data type O.

- IGCANKCNV (Alphanumeric-to-DBCS Conversion)

  This printer file keyword converts alphanumeric SBCS characters to equivalent DBCS characters. Printed SBCS alphanumeric characters have the same appearance as printed DBCS characters. The printed DBCS characters, however, are twice as wide as the equivalent SBCS alphanumeric characters.

- IGCCDEFNT (DBCS Coded Font)

  This printer file keyword specifies the DBCS coded font for printing a named or constant field (or fields).

- IGCCNV (DBCS Conversion)

  This is a display file keyword that enables DBCS conversion.

- IGCCHRRTT (DBCS Character Rotation).

  This printer file keyword rotates each DBCS character 90 degrees counterclockwise before printing. By rotating characters, the system prints them in reading sequence. This keyword should be used only for printer files to be printed with 5553 printers or IPDS AFP(*YES) printers.

For more information on the DDS keywords for DBCS, see the DDS Reference: Concepts topic.

DBCS file data types: The data type of a field in a physical file may be changed when it is being referred to in a logical file, as summarized in the following table:

Physical File Data Types	Logical File Data Types
J	J, O, E, H, G
O	O, H
E	E, O, H
A	A, O, E, H
H	J, O, E, A, H
G	G, O, J, E

The Katakana code page (00290): The Katakana code page (code page 00290) of Japan supports uppercase English and single-byte Katakana (phonetics) characters. The lowercase English characters are located at code points different from other code pages and the hardware may not be able to display English uppercase, lowercase, and Katakana characters concurrently. Therefore, special considerations should be taken if the application is going to support this code page:

- Avoid using the lowercase alphabet for syntactic characters.
- Avoid using the SBCS lowercase alphabet with Japanese DBCS messages.

UCS-2 level-1 support and IBM DBCS displays: OS/400 supports ISO/IEC 10646 Universal Coded Character Set 2, Level 1 (UCS-2, Level-1). IBM DBCS-capable display stations, however, do not support UCS-2 Level-1 data. If you are designing an application to handle UCS-2 Level-1 data for display on an IBM DBCS-capable display, you must convert the data to a mixed-byte CCSID before sending the data to the display station.

Process double-byte characters: Due to the large number of double-byte characters, the system needs more information to identify each double-byte character than is needed to identify each alphanumeric character.

There are two types of double-byte characters: basic and extended. These characters are usually processed by the device on which the characters are displayed or printed.

Basic double-byte characters:

Basic characters are frequently used double-byte characters that reside in the hardware of a DBCS-capable device. The number of double-byte characters stored in the device varies with the language supported and the storage size of the device. A DBCS-capable device can display or print basic characters without using the extended character processing function of the operating system.

Double-byte extended characters:

When processing extended characters, the device requires the assistance of the system. The system must tell the device what the character looks like before the device can display or print the character. Extended characters are stored in a DBCS font table, not in the DBCS-capable device. When displaying or printing extended characters, the device receives them from the DBCS font table under control of the operating system.

Extended character processing is a function of the operating system that is required to make characters stored in a DBCS font table available to a DBCS-capable device.

To request extended character processing, specify the double-byte extended character parameter, IGCEXNCHR(*YES), on the file creation command when you create a display (CRTDSPF) or create a printer file (CRTPRTF) command that processes double-byte data. Because IGCEXNCHR(*YES) is the default value, the system automatically processes extended characters unless you instruct it otherwise. You can change this file attribute by using the change file (CHGDSPF) or (CHGPRTF) command. You can override the file attribute with the override display file (OVRDSPF) or override printerfile (OVRPRTF) command. For example, to override the display file DBCSDSPF so that extended characters are processed, enter:

```
OVRDSPF DSPF(DBCSDSPF) IGCEXNCHR(*YES)
```

Notes:

1. The system ignores the IGCEXNCHR parameter when processing alphanumeric files.
2. When you use the Japanese 5583 Printer to print extended characters, you must use the Kanji print function of the Advanced DBCS Printer Support licensed program.

What happens when extended characters are not processed:

When extended characters are not processed, the following happens:

- Basic double-byte characters are displayed and printed.
- On displays, the system displays the undefined character where it would otherwise display extended characters.
- On printed output, the system prints the undefined character where it would otherwise print extended characters.
- The extended characters, though not displayed or printed, are stored correctly in the system.

Display support: The following provides useful information about displaying double-byte characters.

Inserting shift-control characters:

The system inserts shift-control characters into DBCS-only fields automatically.

To insert shift-control characters into open fields or either fields, do the following:

1. Position the cursor in the field in which you want to insert double-byte data.
2. Press the Insert Shift Control Character key (according to your DBCS display user's guide).

The system inserts a pair of shift-control characters at the same time. The system leaves the cursor under the shift-in character and puts the keyboard in insert mode. Insert double-byte characters between the shift-control characters.

To find out if a field already has the shift-control characters, press the Display Shift Control Character key.

DBCS-graphic fields store double-byte characters without requiring the use of shift control characters. Shift control characters should not be inserted in graphic fields.

Number of displayed extended characters:

The system can display up to 512 different extended characters on a Japanese display at one time. Additional extended characters are displayed as undefined characters. However, the additional extended characters are stored correctly in the system.

Number of input fields on a display:

The use of DBCS input fields affects the total number of input fields allowed on a display. For a local 5250 display, you can specify as many as 256 input fields. However, each three instances of a DBCS field reduces the maximum number of fields by one. For example, if there are 9 DBCS fields on a display, then the maximum is 256 - (9/3) = 253 input fields.

Effects of displaying double-byte data at alphanumeric workstations:

Alphanumeric displays cannot display double-byte data correctly. If you try to display double-byte data at an alphanumeric display, the following happens:

- The system sends an inquiry message to that display, asking whether you want to continue using the program with double-byte data or to cancel it.
- If you continue using the program, the system ignores the shift-control characters and interprets the double-byte characters as though they were single-byte characters. Displayed double-byte data does not make sense.

Make printer files DBCS capable: In many cases, printer files are used by the server to produce data that will eventually be printed or displayed. In these cases, the data is first placed into a spooled file using one of the IBM-supplied printer files. The data is then taken from the spooled file and is displayed or printed based on the request of the user.

When the data involved contains double-byte characters, the printer file that is used to place the data into the spooled file must be capable of processing double-byte data. A printer file is capable of processing double-byte data when *YES is specified on the IGCDTA parameter for the file. In most cases, the system recognizes the occurrence of double-byte data and takes appropriate measures to ensure the printer file that is used is capable of processing double-byte data.

In some cases, however, the system cannot recognize the occurrence of double-byte data and may attempt to use a printer file that is not capable of processing double-byte data. If this occurs, the output at the display or printer may not be readable. This can happen when object descriptions containing double-byte characters are to be displayed or printed on an alphanumeric device.

To ensure that you receive correct results when you display or print double-byte characters, some recommendations should be followed. Action is required on your part if you have a single-byte national language installed as a secondary language. Printer files that are received as part of the DBCS version of a product are always capable of processing DBCS data.

You should complete the following recommended actions after the product or feature has been installed:

1. If all printers and display devices attached to your system are DBCS-capable, you can enable all printer files for double-byte data. For IBM-supplied printer files that are received as part of a single-byte secondary language feature, you can enable all printer files by issuing the following command:

   ```
   CHGPRTF FILE(*ALL/*ALL) IGCDTA(*YES)
   ```

   After this command has been completed, all printer files in all libraries will be enabled for double-byte data. The change will be permanent.
2. If all printer and display devices attached to your system are not DBCS-capable, it is recommended that you do not enable all IBM-supplied printer files.

   Instead, use the library search capabilities of the system to control which printer files will be used for any particular job. When the potential exists that double-byte data will be encountered, the library list for the job should be such that the printer files that are DBCS-enabled will be found first in the library list. Conversely, if only single-byte data is expected to be encountered, the library list should be set up so the printer files that are not enabled for DBCS will be found first. In this way, the printer file capabilities will match the type of data that will be processed. The decision as to what type of printer file to use is made on the basis of what type of data will be processed. The device that will be used to actually display or print the data may also influence this decision.

In some cases it may be desirable to make the printer file only temporarily DBCS-capable instead of making a permanent change. For a specific job, you can make this temporary change by using the OVRPRTF command.

To temporarily enable a specific printer file, you can use the following command:

```
OVRPRTF FILE(filename) IGCDTA(*YES)
```

Where *filename* is the name of the printer file you want to enable.

Copy spooled and nonspooled DBCS files: You can copy both spooled and nonspooled DBCS files.

Copying spooled files

Copy spooled files that have double-byte data by using the Copy Spooled File (CPYSPLF) command. However, the database file to which the file is being copied must have been created with the IGCDTA(*YES) value specified.

When copying spooled files to a database file that contains double-byte data, an extra column is reserved for the shift-out character. This shift-out character is placed between the control information for the record and the user data. The following table shows the shift-out character column number, based on the value specified for the Control character (CTLCHAR) keyword:

CTLCHAR value	Column for shift-out character
*NONE	1
*FCFC	2
*PRTCTL	5
*S36FMT	10

Copying nonspooled DBCS files

You can use the Copy File (CPYF) command to copy double-byte data from one file to another.

When copying data from a double-byte database file to an alphanumeric database file, specify one of the following on the CPYF command:

- If both files are source files or if both files are database files, you can specify either the FMTOPT(*MAP) parameter or the FMTOPT(*NOCHK) parameter.
- If one file is a source file and the other file is a database file, specify the FMT(*CVTSRC) parameter.

When you copy DBCS files to alphanumeric files, the system sends you an informational message describing the difference in file types.

Either the FMTOPT(*MAP) or FMTOPT(*NOCHK) option of the copy file function must be specified for copies from a physical or logical file to a physical file when there are fields with the same name in the from-file and to-file, but the data type for fields is as shown in the following table:

From-file field data type	To-file field data type
A (character)	J (DBCS-only)
O (DBCS-open)	J (DBCS-only)
O (DBCS-open)	E (DBCS-either)
E (DBCS-either)	J (DBCS-only)
J (DBCS-only)	G (DBCS-graphic)
O (DBCS-open)	G (DBCS-graphic)
E (DBCS-either)	G (DBCS-graphic)
G (DBCS-graphic)	J (DBCS-only)
G (DBCS-graphic)	O (DBCS-open)
G (DBCS-graphic)	E (DBCS-either)

When you use FMTOPT(*MAP) on the CPYF command to copy data to a DBCS-only field or DBCS-graphic field, the corresponding field in the from-file must not be:
- Less than a 2-byte character field
- An odd-byte-length character field
- An odd-byte-length DBCS-open field

If you attempt to copy with one of these specified in the from-field, an error message is sent.

When you copy double-byte data from one database file to another with the FMTOPT(*MAP) parameter specified, double-byte data will be copied correctly. The system will perform correct padding and truncation of double-byte data to ensure data integrity.

When using the CPYF command with FMTOPT(*MAP) to copy a DBCS-open field to a graphic field, a conversion error occurs if the DBCS-open field contains any SBCS data (including blanks).

Change alphanumeric programs to DBCS programs: If an alphanumeric application program uses externally described files, you can change that application program to a DBCS application program by changing the externally described files. To convert an application program, do the following:
1. Create a duplicate copy of the source statements for the alphanumeric file that you want to change.
2. Change alphanumeric constants and literals to double-byte constants and literals.
3. Change fields in the file to the open (O) data type or specify the Alternative Data Type (IGCALTTYP) DDS keyword so that you can enter both double-byte and alphanumeric data in these fields. You may want to change the length of the fields as the double-byte data takes more space.
4. Store the converted file in a separate library. Give the file the same name as its alphanumeric version.
5. When you want to use the changed file in a job, change the library list, using the Change Library List (CHGLIBL) command, for the job in which the file will be used. The library in which the DBCS display file is stored is then checked before the library in which the alphanumeric version of the file is stored.

Enter DBCS text in CL commands: You can use double-byte character data anywhere in a CL command that descriptive text can be used.

Enter double-byte character text as follows:

1. Begin the double-byte character text with an apostrophe (').
2. Enter a shift-out character.
3. Enter the double-byte character text.
4. Enter a shift-in character.
5. End the double-byte character text with an apostrophe (').

For example, to enter the double-byte character literal ABC, enter the following, where SO represents the shift-out character and SI represents the shift-in character:

```
'SOABCSI'
```

Limit the length of a double-byte character text description of an object to 14 double-byte characters, plus the shift control characters, to make sure that the description is properly displayed and printed.

DBCS conversion: When you use DBCS displays to enter double-byte data, you may use the various data entry methods supported on the display, or you may choose to use the server DBCS conversion support. DBCS conversion lets you enter an alphanumeric entry or DBCS code and convert the entry or code to its related DBCS word. DBCS conversion is intended for Japanese character sets and its use is limited for application to other double-byte character sets.

Specifically, DBCS conversion lets you convert the following:

- A string of alphanumeric characters to a DBCS word
- English alphanumeric characters to double-byte alphanumeric characters
- Alphanumeric Katakana to double-byte Hiragana and Katakana letters
- A DBCS code to its corresponding double-byte character
- A DBCS number to its corresponding double-byte character

The following links provide additional information about DBCS conversion:

- Conversion dictionaries
- Work with conversion dictionaries
- Japanese DBCS conversion

Conversion dictionaries: The DBCS conversion dictionary is a collection of alphanumeric entries and their related DBCS words. The system refers to the dictionary when performing DBCS conversion.

All DBCS conversion dictionaries have an object type of *IGCDCT. A system-supplied and a user-created dictionary are used with DBCS conversion.

User-created dictionary:

A user-created dictionary contains any alphanumeric entries and related DBCS words that you choose to include. You might create a user dictionary to contain words unique to your business or words that you use regularly but that are not included in the system-supplied dictionary.

You can create one or more DBCS conversion dictionaries with any name and store them in any library. When performing DBCS conversion, however, the system only refers to the first user dictionary named QUSRIGCDCT in the user's library list, no matter how many dictionaries you have or what they are named. Make sure that the library list is properly specified so that the system checks the correct dictionary.

During DBCS conversion, the system checks QUSRIGCDCT before checking QSYSIGCDCT.

DBCS conversion dictionary commands:

You can use the following commands to perform object management functions with the DBCS conversion dictionary. Specify the OBJTYPE(*IGCDCT) parameter when entering these commands:

- CHGOBJOWN: Change the owner of a DBCS conversion dictionary
- CHKOBJ: Check a DBCS conversion dictionary
- CRTDUPOBJ: Create a duplicate object of the dictionary
- DMPOBJ: Dump a DBCS conversion dictionary
- DMPSYSOBJ: Dump the system-supplied dictionary
- DSPOBJAUT: Display a user's authority to the dictionary
- GRTOBJAUT: Grant authority to use the dictionary
- MOVOBJ: Move the dictionary to another library
- RNMOBJ: Rename the dictionary
- RSTOBJ: Restore the dictionary
- RVKOBJAUT: Revoke authority to use the dictionary
- SAVOBJ: Save the dictionary
- SAVCHGOBJ: Save a changed dictionary

The system saves or restores DBCS conversion dictionaries when you use these commands:

- RSTLIB: Restore a library in which the dictionary is stored
- SAVLIB: Save a library in which the dictionary is stored
- SAVSYS: Save QSYSIGCDCT, the system DBCS conversion dictionary, when saving the system

You can use the following commands to create, edit, display, and delete a dictionary:

- CRTIGCDCT: Create DBCS Conversion Dictionary
- EDTIGCDCT: Edit DBCS Conversion Dictionary
- DSPIGCDCT: Display DBCS Conversion Dictionary
- DLTIGCDCT: Delete DBCS Conversion Dictionary

Work with conversion dictionaries: The following topics describe how you create, edit, display, print, and delete conversion dictionaries.

Create a DBCS conversion dictionary

To create a DBCS conversion dictionary, do the following:

1. Use the Create DBCS Conversion Dictionary (CRTIGCDCT) command.
2. Name the dictionary, QUSRIGCDCT, so it can be used during DBCS conversion. The system uses the dictionary if it is the first user-created dictionary found when searching a user's library list.

   You might call the dictionary by another name while it is being created to prevent application programs from using it for conversion. Later, change the dictionary name using the Rename Object (RNMOBJ) command.
3. Use the EDTIGCDCT command to put entries and related words into the dictionary after creating it.

Edit a DBCS conversion dictionary

Use the Edit DBCS conversion dictionary (EDTIGCDCT) command to edit the DBCS conversion dictionary. Use editing to add user-defined characters to the dictionary, so that users can enter characters using DBCS conversion, and rearrange terms in a DBCS conversion dictionary to suit individual needs.

The display needed for use while editing the DBCS conversion dictionary depends on the value that you entered for the ENTRY parameter on the EDTIGCDCT command:

- If you specified a specific string with the ENTRY parameter or if you want to display double-byte characters, you must use a DBCS display.
- If you did not specify a specific string with the ENTRY parameter, or if you do not want to display double-byte characters, use either a DBCS display, or a 24-row by 80-column alphanumeric display.

You may perform the following editing operations on a user-created DBCS conversion dictionary:

- Add entries to the dictionary (including adding the first entries to the dictionary after it is created). The dictionary can contain as many as 99,999 entries.
- Delete entries from the dictionary.
- Change entries in the dictionary, such as replacing the DBCS words related to an alphanumeric entry.
- Move the DBCS words related to an alphanumeric entry to rearrange the order in which they appear during DBCS conversion.

The only editing function that you can perform with QSYSIGCDCT, the system-supplied dictionary, is to move DBCS words related to an alphanumeric entry. Move words in order to rearrange the order in which they appear during DBCS conversion.

Display and print the DBCS conversion dictionary

Use the Display DBCS Conversion Dictionary (DSPIGCDCT) command to display and print the DBCS conversion dictionary. You can display or print the entire dictionary or just a certain part of it, depending on the value you specify for the ENTRY parameter.

For example, to print the entry ABC from the dictionary QUSRIGCDCT and its related words, enter:

```
DSPIGCDCT IGCDCT(DBCSLIB/QUSRIGCDCT)  +
ENTRY(ABC) OUTPUT(*PRINT)
```

To display all of the entries from the system-supplied dictionary QSYSIGCDCT and their related words, enter:

```
DSPIGCDCT IGCDCT(QSYS/QSYSIGCDCT)
```

Delete a DBCS conversion dictionary

Use the Delete DBCS Conversion Dictionary (DLTIGCDCT) command to delete a DBCS conversion dictionary from the system. In order to delete the dictionary, you must have object existence authority to the dictionary and object operational authorities to the library in which the dictionary is stored.

When you delete a dictionary, make sure that you specify the correct library name. It is possible that many users have their own dictionaries, each named QUSRIGCDCT, stored in their libraries. If you do not specify any library name, the system deletes the first DBCS conversion dictionary in your library list.

Japanese DBCS conversion: When you use DBCS displays to enter double-byte data, you may use the various data entry methods supported on the display, or you may choose to use the OS/400 DBCS conversion support. DBCS conversion lets you enter an alphanumeric entry or DBCS code and convert the entry or code to its related DBCS word. DBCS conversion is intended for Japanese character sets and its use is limited for application to other double-byte character sets.

Specifically, DBCS conversion lets you convert the following:

- A string of alphanumeric characters to a DBCS word
- English alphanumeric characters to double-byte alphanumeric characters
- Alphanumeric Katakana to double-byte Hiragana and Katakana letters

- A DBCS code to its corresponding double-byte character
- A DBCS number to its corresponding double-byte character

Japanese system-supplied dictionary

The QSYSIGCDCT is the system-supplied dictionary that is stored in the library, QSYS. It is a collection of entries with a Japanese pronunciation, expressed in alphanumeric characters, and the DBCS words related to those entries. The system checks this dictionary second when performing DBCS conversion.

QSYSIGCDCT contains these entries:

- Personal names
  - Family names
  - First names
- Organization names
  - Private enterprises registered in the security market
  - Public corporations
  - Typical organizations in the central and local governments
  - Most universities and colleges
- Addresses
  - Public administration units within the prefectures
  - Towns and streets in 11 major cities
- Business terms, such as department names and position titles commonly used in enterprises
- Individual double-byte characters, including basic double-byte characters, as defined by IBM

You cannot add or delete entries from this dictionary. However, you may rearrange the related DBCS words so that the words used most frequently are displayed first during DBCS conversion.

SQL and DBCS: The basic symbols of keywords and operators in the SQL language are single-byte characters that are part of all character sets supported by the IBM relational database products. Characters of the language are classified as letters, digits, or special characters.

SQL host identifiers and double-byte characters

A host-identifier is a name declared in the host program. The rules for forming a host-identifier are the rules of the host language, except that DBCS characters cannot be used.

SQL character subtypes and double-byte characters

Each character string is further defined as one of the following:

- **Bit data:** Data that is not associated with a coded character set and is never converted. The CCSID for bit data is 65535.
- **SBCS data:** Data in which every character is represented by a single byte. Each SBCS data character string has an associated CCSID. If necessary, an SBCS data character string is converted before it is used in an operation with a character string that has a different CCSID.
- **Mixed data:** Data that may contain a mixture of characters from a single-byte character set (SBCS) and a double-byte character set (DBCS). Each mixed data character string has an associated CCSID. If necessary, a mixed data character string is converted before an operation with a character string that has a different CCSID. If mixed data contains a DBCS character, it cannot be converted to SBCS data.

The database manager does not recognize subclasses of double-byte characters, and it does not assign any specific meaning to particular double-byte codes. However, if you choose to use mixed data, then two single-byte EBCDIC codes are given special meanings:

- X'0E', the "shift-out" character, is used to mark the beginning of a sequence of double-byte codes.
- X'0F', the "shift-in" character, is used to mark the end of a sequence of double-byte codes.

In order for the database manager to recognize double-byte characters in a mixed data character string, the following condition must be met:

- Within the string, the double-byte characters must be enclosed between paired shift-out and shift-in characters.

  The pairing is detected as the string is read from left to right. The code X'0E' is recognized as a shift out character if X'0F' occurs later; otherwise, it is invalid. The first X'0F' following the X'0E' that is on a double-byte boundary is the paired shift-in character. Any X'0F' that is not on a double-byte boundary is not recognized.

  There must be an even number of bytes between the paired characters, and each pair of bytes is considered to be a double-byte character. There can be more than one set of paired shift-out and shift-in characters in the string.

The length of a mixed data character string is its total number of bytes, counting two bytes for each double-byte character and one byte for each shift-out or shift-in character.

When the job CCSID indicates that DBCS is allowed, CREATE TABLE will create character columns as DBCS-Open fields, unless FOR BIT DATA, FOR SBCS DATA, or an SBCS CCSID is specified. The SQL user will see these as character fields, but the system database support will see them as DBCS-Open fields.

For more information on SQL and DBCS, see the following:

- SQL graphic strings
- SQL assignments and comparisons
- SQL conversion rules

SQL graphic strings: A graphic string is a sequence of double-byte characters that does not include shift-out or shift-in characters. The length of the string is the number of its characters. Like character strings, graphic strings can be empty.

Every graphic string has a CCSID that identifies a double-byte coded character set. If necessary, a graphic string is converted before it is used in an operation with a graphic string that has a different CCSID.

SQL fixed-length and double-byte characters

All values of a fixed-length graphic-string column have the same length, which is determined by the length attribute of the column. The length attribute must be between 1 through 16383 inclusive.

SQL graphic-string constants

A graphic-string constant is a varying-length graphic string. The length of the specified string cannot be greater than 16370.

In the normal form, the SQL delimiters and the G or the N are SBCS characters. The SBCS apostrophe (') is the EBCDIC apostrophe, X'7D'.

In the PL/I form, the apostrophes and the G are DBCS characters. Two consecutive DBCS string delimiters are used to represent one string delimiter within the string. Notice that this PL/I form is only valid for static statements embedded in PL/I programs.

A hexadecimal graphic constant is also supported. The form of the hexadecimal graphic constant is:

```
GX'ssss'
```

In the constant, **ssss** represents a string from 0 to 32766 hexadecimal digits. The number of characters between the string delimiters must be an even multiple of 4. Each group of 4 digits represents a single graphic character. The hexadecimal for shift-in and shift-out (X'0E' and X'0F') are not included in the string.

The CCSID assigned to constants is the DBCS CCSID associated with the CCSID of the source unless the source is encoded in a foreign encoding scheme (such as ASCII). In this case, the CCSID assigned to the constant is the DBCS CCSID associated with the default CCSID of the application server when the SQL statement containing the constant is prepared. If there is no DBCS CCSID associated with the CCSID of the source, the CCSID is 65535.

SQL assignments and comparisons: The basic operations of SQL are assignment and comparison. Assignment operations are performed during the running of CALL, INSERT, UPDATE, FETCH, and SELECT INTO statements. Comparison operations are performed during the running of statements that include predicates and other language elements such as MAX, MIN, DISTINCT, GROUP BY, and ORDER BY.

The basic rule for both operations is that the data type of the operands involved must be compatible. The compatibility rule also applies to UNION, concatenation, and the VALUE, COALESCE, MIN, and MAX scalar functions.

SQL string assignments and double-byte characters

The basic rule for string assignments is that the length of a string assigned to a column must not be greater than the length attribute of the column. (Trailing blanks are normally included in the length of the string. For string assignment, however, trailing blanks are not included in the length of the string.)

If the string contains mixed data, the assignment rules may require truncation within a sequence of double-byte codes. To prevent the loss of the shift-in character that ends the double-byte sequence, additional characters may be truncated from the end of the string, and a shift-in character added. In the truncated result, there is always an even number of bytes between each shift-out character and its matching shift-in character.

Character, DBCS-only, DBCS-open, and DBCS-either are not compatible with graphic types for assignment.

SQL conversion rules: When two strings are compared, one of the strings is first converted, if necessary, to the coded character set of the other string. Character conversion is necessary only if all of the following are true:

- The CCSIDs of the two strings are different.
- Neither CCSID is 65535.
- The string selected for conversion is neither null nor empty.
- The CCSID conversion selection table indicates that conversion is necessary.

If one string has an SBCS CCSID and the other is the same type of operand and has a mixed data CCSID, the SBCS data character string is converted. Otherwise, the string selected for conversion depends on the type of each operand. The following table shows which operand is selected for conversion, given the operand types.

First operand	Column value (second operand)	Derived value (second operand)	Special register (second operand)	Constant (second operand)	Host variable (second operand)
Column value	Second	Second	Second	Second	Second
Derived Value	First	Second	Second	Second	Second

First operand	Column value (second operand)	Derived value (second operand)	Special register (second operand)	Constant (second operand)	Host variable (second operand)
Special Register	First	First	Second	Second	Second
Constant	First	First	First	Second	Second
Host Variable	First	First	First	First	Second

A host variable containing data in a foreign encoding scheme is always effectively converted to the native encoding scheme before it is used in any operation. The above rules are based on the assumption that this conversion has already occurred.

An error occurs if a character of the string cannot be converted or the CCSID Conversion Selection Table is used but does not contain any information about the pair of CCSIDs. A warning occurs if a character of the string is converted to the substitution character.

DBCS code schemes

IBM supports two DBCS code schemes: one for the host systems, the other for personal computers. The DBCS code scheme for host systems has the following code-range characteristics:

- First byte: hex 41 to hex FE
- Second byte: hex 41 to hex FE
- Double-byte blank: hex 4040

Shift-control characters

When the host code scheme is used, the system uses shift-control characters to identify the beginning and end of a string of double-byte characters. The shift-out (SO) character, hex 0E, indicates the beginning of a double-byte character string. The shift-in (SI) character, hex 0F, indicates the end of a double-byte character string.

Each shift-control character occupies the same amount of space as one alphanumeric character. By contrast, double-byte characters occupy the same amount of space as two alphanumeric characters.

When double-byte characters are stored in a graphic field or a variable of graphic data type, there is no need to use shift control characters to surround the double-byte characters.

Incorrect and undefined double-byte code

Incorrect double-byte code has a double-byte code value that is not in the valid double-byte code range. This is in contrast to undefined double-byte code where the double-byte code is valid, but no graphic symbol has been defined for the code.

Supported DBCS code ranges

OS/400 supports the following DBCS character-set code ranges:

- Japanese character-set code range
- Korean character-set code range
- Simplified Chinese character-set code range
- Traditional Chinese character-set code range

See Appendix A (DBCS Code Scheme) in the File Management PDF for details.

DBCS font tables

DBCS font tables contain the images of the double-byte extended characters used on the system. The system uses these images to display and print extended characters

when they are not resident on the device.

The following DBCS font tables are objects that you can save or restore. These font tables are distributed with the DBCS national language versions of the OS/400 licensed program:

QIGC2424
A Japanese DBCS font table used to display and print extended characters in a 24-by-24 dot matrix image. The system uses the table with Japanese displays, printers attached to displays, 5227 Model 1 Printer, and the 5327 Model 1 Printer.

QIGC2424C
A Traditional Chinese DBCS font table used to print extended characters in a 24-by-24 dot matrix image. The system uses the table with the 5227 Model 3 Printer and the 5327 Model 3 Printer.

QIGC2424K
A Korean DBCS font table used to print extended characters in a 24-by-24 dot matrix image. The system uses the table with the 5227 Model 2 Printer and the 5327 Model 2 Printer.

QIGC2424S
A Simplified Chinese DBCS font table used to print extended characters in a 24-by-24 dot matrix image. The system uses the table with the 5227 Model 5 Printer.

QIGC3232
A Japanese DBCS font table used to print characters in a 32-by-32 dot matrix image. The system uses the table with the 5583 Printer and the 5337 Model 1 Printer.

QIGC3232S
A Simplified Chinese DBCS font table used to print characters in a 32-by-32 dot matrix image. The system uses the table with the 5337 Model R05 Printer.

All DBCS font tables have an object type of *IGCTBL. You can find instructions for adding user-defined characters to DBCS font tables in the *ADTS/400: Character Generator Utility*, SC09-1769-00 book.

DBCS font table commands

The following commands allow you to manage and use DBCS font tables:

- Check DBCS Font Table (CHKIGCTBL)
- Copy DBCS Font Table (CPYIGCTBL)
- Delete DBCS Font Table (DLTIGCTBL)
- Start Font Management Aid (STRFMA)

Locate an existing font table

Use the Check DBCS Font Table (CHKIGCTBL) command to find out if a DBCS font table exists in your system.

For example, to find out if the table QIGC2424 exists, enter:

```
CHKIGCTBL IGCTBL(QIGC2424)
```

If the table does not exist, the system responds with a message. If the table does exist, the system simply returns without a message.

Check for the existence of a table when adding a new type of DBCS workstation to make sure that the table used by the device exists in the system.

For more information

For additional information, see the following:

- Copy a DBCS font table
- Delete a DBCS font table

Copy a DBCS font table: Use the Copy DBCS Font Table (CPYIGCTBL) command to copy a DBCS font table to or from tape, diskette, or physical file.

The DBCS font tables are saved when you use the Save System (SAVSYS) command so you do not have to use the CPYIGCTBL command when performing normal system backup.

A physical file used to save and restore table information must have a minimum record length of 74 bytes.

Copying a table onto a tape, a diskette, or a physical file

You should copy a DBCS font table onto a tape, a diskette, or a physical file in the following instances:

- Before deleting that table
- After new user-defined characters are added to the tables
- When planning to use the tables on another system

To copy a DBCS font table onto a tape, a diskette, or a physical file do the following:

1. If copying a DBCS font table onto a tape or diskettes, make sure that the tape or diskettes are initialized to the *DATA format. If necessary, initialize the tape or diskettes by specifying the FMT(*DATA) parameter on the Initialize Diskette (INZDKT) command.
2. Load the initialized tape or diskette onto the system.
3. Enter the CPYIGCTBL command as follows:
   a. Choose the value OPTION(*OUT).
   b. Use the DEV parameter to select the device to which you want to copy the table. A value of *FILE specifies that the DBCS font table is saved to a physical file.
   c. Use the SELECT and RANGE parameters to specify which portion of the table you want copied from the system. For more information on the valid codes and numbers to specify for starting and ending values of user-defined character ranges, see the .
4. Press the Enter key. The system copies the DBCS font table onto the specified medium or into a physical file.
5. Remove the tape or the diskette after the system finishes copying the table.

Copying a DBCS font table from a tape, a diskette, or a physical file

Use the Copy DBCS Font Table (CPYIGCTBL) command to copy a DBCS font table from a tape, a diskette, or a physical file onto the system. The system automatically creates the DBCS font table again when copying its contents if the following are true:

- The specified table does not already exist in the system.
- The medium or physical file from which you are copying the table contains all of the IBM-defined double-byte characters.
- SELECT(*ALL) or SELECT(*SYS) is specified on the CPYIGCTBL command.

Delete a DBCS font table: Use the Delete DBCS Font Table (DLTIGCTBL) command to delete a DBCS font table from the server.

Delete an unused DBCS font table to free storage space. For example, if you do not plan to use Japanese printer 5583 or 5337 with your server, font table QIGC3232 is not needed and can be deleted.

When deleting a table, do the following:

1. If desired, copy the table onto a tape, a diskette, or a physical file. If you do not copy the table before deleting it, you will not have a copy of the table for future use.
2. Vary off all devices using that table.
3. Enter the DLTIGCTBL command. For example, to delete the DBCS font table QIGC3232, enter:
   ```
   DLTIGCTBL IGCTBL(QIGC3232)
   ```
4. Press the Enter key. The system sends an inquiry message to the system operator message queue for you to confirm your intention to delete a DBCS table.
5. Respond to the inquiry message. The server sends you a message when it has deleted the table.

   Note: Do not delete a DBCS font table if any device using that table is currently varied on. Also, make sure that the affected controller is not varied on. If you try to delete the table while the device and controller are varied on, the system reports any devices attached to the same controller(s) as those devices, and the controller(s) as damaged the next time you try to print or display extended characters on an affected device. If such damage is reported, do the following:

   a. Vary off the affected devices, using the Vary Configuration (VRYCFG) command.
   b. Vary off the affected controller.
   c. Vary on the affected controller.
   d. Vary on the affected devices.
   e. Continue normal work.

DBCS font files

In addition to the system-supplied DBCS font tables, the system also provides DBCS font files. These DBCS font files are physical files which contain frequently used double-byte characters. When using the character generator utility, you can use the characters in these files as the base for a new user-defined character. These files are supplied with read-only authority as they are not to be changed. If you do not use character generator utility or the Advanced DBCS Printer Support licensed program, you may delete these files to save space. They all exist in the QSYS library.

The following DBCS font files are distributed with the DBCS national language versions of the OS/400 licensed program. They are used as a reference for the CGU and the Advanced DBCS Printer Support licensed program.

QCGF2424
A Japanese DBCS font file used to store a copy of the Japanese DBCS basic character images.

QCGF2424K
A Korean DBCS font file used to store a copy of the Korean DBCS basic character images.

QCGF2424C
A Traditional Chinese DBCS font file used to store a copy of the Traditional Chinese DBCS basic character images.

QCGF2424S
A Simplified Chinese DBCS font file used to store a copy of the Simplified Chinese DBCS basic character images.

DBCS sort tables

DBCS sort tables contain the sort information and collating sequences of all the double-byte characters used on the system. The sort utility on the system uses these tables to sort double-byte characters.

DBCS sort tables are objects that you can save, restore and delete. Using the character generator utility you can also add, delete and change entries in these tables corresponding to the image entries in the DBCS font tables. For Japanese use only, you can also copy the DBCS master sort table to and from a data file.

The following DBCS sort tables are distributed with the DBCS national language versions of OS/400 licensed program:

QCGMSTR
A Japanese DBCS master sort table used to store the sort information for the Japanese double-byte character set.

QCGACTV
A Japanese DBCS active sort table used to store the sort collating sequences for the Japanese double-byte character set.

QCGMSTRC
A Traditional Chinese DBCS master sort table used to store the sort information for the Traditional Chinese double-byte character set.

QCGACTVC
A Traditional Chinese DBCS active sort table used to store the sort collating sequences for the Traditional Chinese double-byte character set.

QCGACTVK
A Korean DBCS active sort table used to map Hanja characters to Hangeul characters with equivalent pronunciation.

QCGMSTRS
A Simplified Chinese DBCS master sort table used to store the sort information for the Simplified Chinese double-byte character set.

QCGACTVS
A Simplified Chinese DBCS active sort table used to store the sort collating sequences for the Simplified Chinese double-byte character set.

You can sort Japanese, Korean, Simplified Chinese, and Traditional Chinese double-byte characters. Each of these languages have two DBCS sort tables, a DBCS master sort table and a DBCS active sort table, except for Korean which has only a DBCS active sort table. The DBCS master sort table contains sort information for all defined DBCS characters. The DBCS active sort table for Japanese, Simplified Chinese, and Traditional Chinese is created from the master sort table information and contains the collating sequences for the double-byte characters of that given language. These collating sequences have a purpose similar to the EBCDIC and ASCII collating sequences for the single-byte alphanumeric character set. For Korean characters, the Hangeul characters are assigned both their collating sequence as well as their DBCS codes according to their pronunciation. Hence, a separate collating sequence is not required, and each of the Hanja characters is mapped to a Hangeul character of the same pronunciation using the DBCS active sort table QCGACTVK.

All DBCS sort tables have an object type of *IGCSRT.

Commands for DBCS sort tables

The following commands allow you to manage and use DBCS sort tables.

- Check Object (CHKOBJ)
- Save Object (SAVOBJ)
- Restore Object (RSTOBJ)

Use existing DBCS sort tables

You can save the tables to tape or diskette, delete them from the server, and restore them to the server. The Japanese DBCS master sort table can also be copied to a data file and copied from a data file so that it can be shared with an Application System/Entry* (AS/Entry) system. You can also add sort information for each user-defined character, and add that character to the DBCS collating sequence, as you create it using the character generator utility.

Find existing DBCS sort table

Use the Check Object (CHKOBJ) command to find out if a DBCS sort table exists in your system.

For example, to find out if the table QCGMSTR exists, enter:

```
CHKOBJ OBJ(QSYS/QCGMSTR) OBJTYPE(*IGCSRT)
```

If the table does not exist, the system responds with a message. If the table does exist, the system simply returns without a message.

Check for the existence of a DBCS active sort table when you want to sort double-byte characters for the first time. The DBCS active table for the DBCS language must exist to sort the characters.

For additional information about DBCS sort tables, see the following:

- Save and restore a DBCS sort table
- Delete a DBCS sort table

Save and restore a DBCS sort table: The following topics describe how you save and restore DBCS sort tables.

Save a DBCS sort table to tape or diskette

Save a DBCS sort table onto tape or diskette in the following instances:

- Before deleting that table
- After information is added, updated, or changed in the tables using the character generator utility
- When planning to use the tables on another iSeries server

Use the Save Object (SAVOBJ) command to save a DBCS sort table onto tape or diskette. Specify *IGCSRT for the object type.

The DBCS sort tables are saved when you use the SAVSYS command so you do not have to use the SAVOBJ command when performing normal system backup.

Restore a DBCS sort table from tape or diskette

Use the RSTOBJ command to restore a DBCS sort table from a tape or a diskette onto the system. The tables on the tape or diskette must previously have been saved using the SAVOBJ command. Specify *IGCSRT for the object type. The system automatically re-creates the DBCS sort table when the specified table does not already exist in the system.

These tables must be restored to the QSYS library for the system to know they exist. For that reason, RSTOBJ restores *IGCSRT objects only to the QSYS library and only if the objects do not already exist there.

Delete a DBCS sort table: Use the DLTIGCSRT command to delete a DBCS sort table from the system.

You can delete an unused DBCS sort table to free disk space, but you should always first save a copy of the table using the SAVOBJ command. You should delete the DBCS master sort table for a DBCS language if either of the following are true:

- You will not be creating any new characters for that language using the character generator utility.
- You will not be using the sort utility to sort characters for that language.

You should delete the DBCS active sort table for a DBCS language if you will not be using the sort utility to sort characters for that language. The DBCS active sort table must be on the system to use the sort utility for this language.

When deleting a table, do the following:

1. If desired, save the table onto tape or diskettes. If you do not save the table onto removable media before deleting it, you will not have a copy of the table for future use.
2. Enter the DLTIGCSRT command. For example, to delete the DBCS sort table QCGACTV, enter:
   ```
   DLTIGCSRT IGCSRT(QCGACTV)
   ```
3. Press the Enter key. The system sends you a message when it has deleted the table.

DBCS field definition

Consider the characteristics of DBCS data when defining a DBCS field:

- Each DBCS character is 2 bytes long.
- The length of a DBCS character string is always even.
- Shift-out (SO) and shift-in (SI) control characters are required at the beginning and end of the DBCS character string, except for graphic-data type fields. Together, these characters are 2 bytes long.
- The system treats DBCS data the same as character data, and therefore cannot perform arithmetic operations on it.
- The following DBCS data types can be used to identify DBCS fields:
  - J (DBCS-only) for fields that can contain only bracketed DBCS data.
  - E (DBCS-either) for fields that can contain bracketed DBCS or SBCS data, but not both.
  - O (DBCS-open) for fields that can contain both SBCS and bracketed DBCS data.
  - G (DBCS-graphic) for fields that can contain graphic data without the SO and SI control characters.

  Note: Data type O is allowed in all types of files. Data types J and E are allowed only in database and display files. Data type G is allowed in database, display, and printer files. In most cases, the OS/400 automatically inserts shift-out and shift-in characters. An exception is when data is written into a data type G field in a database file.

  For more information on the DBCS data types, see the DDS Reference: Concepts topic.

Work with locales

Locales are used primarily in ILE-based application programs. Additionally, the Retrieve Locale Information (OPM, QLGRTVLC; ILE, QlgRetrieveLocaleInformation) API retrieves one or all categories of a locale. See the OS/400 API topic for more information.

Benefits of using locales in global applications

Applications can be created independent of language, cultural data, or specific characters. Locales can be accessed to provide this type of support to any integrated language environment-based application.

For example, the LC_TIME category within a locale can be defined in any of the following ways, or in any combination that is convenient for the environment in which the application runs:

- HH:MM:SS
- MM:SS:HH
- SS:MM:HH

Creating locales

Locales are created using the Create Locale (CRTLOCALE) command.

The source file used to create the locale is named QLOCALESRC, in the QSYSLOCALE library. This library is loaded with option 21 of the operating system.

These source files cannot be changed. Instead, they must be copied and then edited if changes are desired.

For a list of source definition files, see System-supplied locales and recommended CCSIDs. To see how to use the CRTLOCALE command, see “Example: Creating a locale” on page 152.

Working with locales

The following topics provide more detailed information about how you can use locales:

- Locale restrictions
- Locale categories
- Locale symbolic names
- Examples: Locale programming

For more information

The following links provide additional information about locales:

- Locales
- Install and enable locales
- System-supplied locales and recommended CCSIDs

Locale restrictions

The following list contains restrictions when using locales to set job attributes:

- The locale CCSID must be an EBCDIC single-byte CCSID for an SBCS system.
- The locale CCSID must be an EBCDIC, single-byte character set (SBCS), or mixed-byte CCSID for a DBCS-capable system.
- The locale object must exist in the QSYS file system.
- The DATFMT, DATSEP, TIMSEP, and DECFMT parameters within the locale must be valid values supported as job attributes. See the Work Management topic for more information on jobs and their attributes.
- If you want sort sequence support from the locale, you must use the CPYSYSCOL keyword. See CPYSYSCOL for more detailed information.

Locale categories

The following categories are supported on OS/400.

Locale category	Description
LC_COLLATE	Defines character or string collation information
LC_CTYPE	Defines character classification, case conversion, and other character attributes.
LC_MESSAGES	Defines the format for affirmative and negative responses.
LC_MONETARY	Defines rules and symbols for formatting monetary numeric information.
LC_NUMERIC	Defines a list of rules and symbols for formatting non-monetary numeric information.
LC_TIME	Defines a list of rules and symbols for formatting time and date information.
LC_TOD	Defines rules for daylight savings time and time zone information.

Note: A locale source file cannot contain duplicate categories.

Locale category source definitions:

The category source definition consists of:

- The category header (*category name*), where the category name must be all uppercase characters.
- The associated keyword/value pairs that comprise the category body. Keywords may be all uppercase, all lowercase, or mixed case characters.
- The category trailer (which consists of END *category-name*)

For example:

```
LC_CTYPE
source for LC_CTYPE category
END LC_CTYPE
```

Lines preceding the first category header can be used to change the comment character and the escape character. The comment_char (the default is #) and escape_char (the default is \) keywords can be used to change these characters. The following example shows how to change the comment character and escape character to * and / respectively:

```
comment_char <asterisk>
escape_char  <slash>
```

Note: This example uses symbolic names to represent the '*' and '/' characters.

The source for all categories is specified using the following:

Keywords

Each keyword identifies either a definition or a rule. The remainder of the statement containing the keyword contains the operands to the keyword. Operands are separated from the keyword by one or more blank characters. A statement may be continued on the next line by placing an escape_char as the last character before the newline or linefeed character that ends the line.

Lines containing the comment_char in the first column are treated as comment lines. Comment lines cannot be continued on a subsequent line using an escape character. \ is the default escape character. However, the escape character can be defined to be any character by the user.

Strings

Strings must be enclosed in double-quotes. Double quotes within strings can be represented in two ways:

- Literally. The escape character can be followed by double quotes.
- A symbolic name. For example, <quotation-mark>.

A string can be continued on the next line by placing an escape_char as the last character before the newline or linefeed character that ends the line.

A string is a sequence of character symbols, or literals enclosed by double-quotation ("") characters. For example:

```
"<A><B><C>"
```

Character literals

A character literal is the character itself.

Character symbols

A character symbol begins with the < (less-than) character, followed by non-control characters, and ends with the > (greater-than) character. For example, <A> is a valid character symbol (symbolic name). Any character symbol referenced in the source file should be one of the

predefined system-supplied symbols. The system supplied symbolic names are in the source file member QLGSYMBOL in the QLOCALESRC source file in the optionally installable library QSYSLOCALE.

See System-supplied locale source definition files for a list of all system-supplied symbolic names.

In the event that the system does not contain a predefined symbolic name for a character, the UCS-2 level-1 format is allowed. The UCS-2 format is based on the character set defined in ISO/IEC 10646. The UCS-2 format may also be used in place of the predefined symbolic names. The following is an example of the UCS-2 symbolic name format:

```
<Uxxxx>
```

where 'xxxx' are four hexadecimal digits. For example, <U0041>. The hexadecimal number 0041 within this symbolic name is the UCS-2 code point that represents the character 'A'.

Each category must be explicitly defined in a locale definition source file is required.

See the Example: POSIX locale for a complete description of each locale category included in the POSIX locale.

LC_COLLATE category: The LC_COLLATE category defines character or string collation information. Within LC_COLLATE you can specify a sort sequence to use using the cpysyscol keyword. The cpysyscol keyword value is used in place of the LC_COLLATE category definitions.

A collation element is the unit of comparison for collation. A collation element may be a character or a sequence of characters. Every collation element in the locale has a set of weights, which determine if the collation element collates before, equal to, or after the other collation elements in the locale. Each collation element is assigned collation weights by the CRTLOCALE command when the locale definition source file is created. These collation weights are then used by applications programs that compare strings.

Every character defined in the CCSID that is specified in the CRTLOCALE command is itself a collating element. Additional collating elements can be defined using the collating-element statement. The syntax is:

```
collating-element character-symbol from string
```

The LC_COLLATE category begins with the LC_COLLATE keyword and ends with the END LC_COLLATE keyword.

The following keywords are recognized in the LC_COLLATE category:

cpysyscol

This statement specifies that a system collating sequence table is to be used for the collation information for the category. If the locale is intended to be used to set the sort sequence table for the job, then it is required that the CPYSYSCOL keyword be used. If the CPYSYSCOL keyword is specified, no other keyword may be specified. The syntax for the CPYSYSCOL keyword is:

```
CPYSYSCOLsort sequence path name;langid
```

The *sort sequence path name* is a string specifying a fully expanded path name of an existing sort sequence table to use as the definition for this category. The path name delimiter must be a slash (/). Other valid values are strings containing one of the following:

***JOB** The sort sequence of the job.

***LANGIDUNQ**

The unique-weighted sort sequence table that is associated with the language identifier requested parameter.

***LANGIDSHR**

The shared-weighted sort sequence table that is associated with the language identifier requested parameter.

***HEX** The sort sequence according to the hexadecimal value of the characters.

The *langid* is a string specifying the language identifier of the sort sequence table to be used. All langids must be in uppercase. Valid values are strings containing one of the following:

***JOB** Use the language identifier of the job.

language id
A valid 3-character language identifier. For example, Danish would be DAN. See Language identifiers and associated default CCSIDs for a complete list of valid language identifiers.

Collating-element

The collating-element statement specifies multi-character collating elements. The syntax for the collating-element statement is:

```
collating-element symbolic-name from string
```

The symbolic-name value defines a collating element that is a string of one or more characters as a single collating element. The symbolic-name value cannot duplicate any system predefined symbolic name, or any other symbolic name defined in this collation definition. The string value specifies a string of two or more characters or character symbols that define the symbolic-name value. The following are examples of the syntax for the collating-element statement:

```
collating-element <ch> from "<c><h>"
collating-element <e-acute> from "<acute><e>"
collating-element <11> from "<1><1>"
```

A symbolic-name value defined by the collating-element statement is recognized only with the LC_COLLATE category.

Order_start

The order_start statement may be followed by one or more collation order statements, assigning collation weights to collating elements. This statement is required. The syntax for the order_start statement is:

```
order_start  sort-rules;sort-rules;...sort-rules collation-order-statements order_end
```

The sort-rules have the following syntax:

```
directive, directive,...directive
```

where directive is one of the directives; **forward, backward,** and **position.**

The sort-rules directives are optional. If present, they define the rules to apply during string comparison. The number of specified sort-rules directives defines the number of weights each collating element is assigned (that is, the number of collation orders in the locale). If no sort-rules directives are present, one forward directive is assumed.

If present, the first sort-rules directive applies when comparing strings using primary weight, the second when comparing strings using the secondary weight, and so on. Each set of sort-rules directives is separated by a ; (semicolon). A sort-rules directive consists of one or more comma-separated directives. The following directives are supported:

Forward
Specifies that collation weight comparisons proceed from the beginning of a string toward the end of the string.

Backward
Specifies that collation weight comparisons proceed from the end of a string toward the beginning of the string.

Position

Specifies that collation weight comparisons consider the relative position of non-ignored elements in the string. That is, if strings compare equal, the element with the shortest distance from the starting point of the string collates first.

The forward and backward directives are mutually exclusive. The following is an example of the syntax for the sort-rules directives:

```
order_start  forward;backward
```

Order_end

This keyword ends collating order entries introduced by the order_start keyword.

The order of the characters and elements specified between the order_start and order_end keywords defines the character order used in range expressions and regular expressions. If no weights are assigned to the characters, then the character order also becomes the collation sequence weight.

Special symbols

Special symbols are required to be all upper case characters. The following special symbols can be used in the LC_COLLATE category:

- IGNORE

  The optional operands for each collation element are used to define the primary, secondary, or subsequent weights for the collating element. The special symbol IGNORE is used to indicate a collating element that is to be ignored when strings are compared.
- UNDEFINED

  All characters in the character set must be placed in the collation order, either explicitly or implicitly, by using the Undefined symbol. The UNDEFINED symbol includes all coded character set values not specified explicitly. These characters are inserted in the character collation order at the point indicated by the Undefined symbol in the order of their character code page values. If a collating weight is not explicitly specified for the UNDEFINED symbol, then by default, all of the undefined characters are assigned the same collating weight equal to the relative order of the first undefined character in the collating sequence. If no UNDEFINED special symbol exists and the collation order does not specify all collation elements from the coded character set, a warning is issued and all undefined characters are placed at the end of the character collation order and be given the same collating weight.

Example 1:

The following is an example of a collation order statement in the LC_COLLATE locale definition source file category.

The text below the LC_COLLATE keywords has been added for clarity and does not appear in the locale source file.

```
order_start  forward;backward
#           The order_start has two sort rules specified:
#           forward and backward

UNDEFINED   IGNORE;IGNORE
#           The UNDEFINED special symbol indicates that
#           all characters in the CCSID of the locale
#           that are not specified in the definition
#           are ignored for collation purposes.

<LOW>
#           <LOW> is a collating symbol that is ordered
#           after all undefined characters.  For example, if there
```

```
#           were only two undefined characters, then the <LOW> symbol
#           would be third in the order.

#           All collating elements between <space> and <a> have the
#           same primary equivalence class and individual secondary
#           weights based on their coded character set values.

<a>      <a>;<a>
<a-acute> <a>;<a-acute>
<a-grave> <a>;<a-grave>
<A>       <a>;<A>
<A-acute> <a>;<A-acute>
<A-grave> <a>;<A-grave>
#           All characters between <a> and <A-grave> belong to the
#           same primary equivalence class because they have the same
#           primary weight.

<ch>      <ch>;<ch>
<Ch>      <ch>;<Ch>
#           The <c><h> multi-character collating element is
#           represented by the <ch> collating symbol and belongs to the
#           same primary equivalence class as the <Ch> multi-character
#           collating element.

<s>       <s>;<s>
<eszet>   "<s><s>";<s>
#           A one-to-many mapping is indicated by the <eszet>
#           character collated as an <s><s> string.  That is, one
#           <eszet> character is expanded to <s><s> characters before
#           comparing.

<HIGH>
order_end
```

Example 2:

Following is an example of a CPYSYSCOL statement in the LC_COLLATE locale definition source file category.

```
LC_COLLATE

CPYSYSCOL "//QSYS.LIB//QLA10025S.TBL";"ENU"

END LC_COLLATE
```

LC_CTYPE category: The LC_CTYPE category defines character classification, case conversion, and other character attributes.

The LC_CTYPE category begins with an LC_CTYPE category header and ends with an END LC_CTYPE category trailer.

All operands for LC_CTYPE category statements are defined as lists of characters. Each list consists of one or more semicolon-separated characters or symbolic character names.

The following keywords are recognized in the LC_CTYPE category. In the descriptions, the term *automatically included* means that an error does not occur if the referenced characters are included or omitted. The characters are provided if they are missing and are accepted if they are present. In the event that the *automatically included* characters do not exist in the CCSID that you want to create the locale, a warning is issued by the CRTLOCALE command.

upper Defines uppercase letter characters. No character defined by the cntrl, digit, punct, or space keyword can be specified. At a minimum, the uppercase letters A through Z are automatically included.

lower Defines lowercase letter characters. No character defined by the cntrl, digit, punct, or space keyword can be specified. At a minimum, the lowercase letters a through z are automatically included.

alpha Defines all letter characters. No character defined by the cntrl, digit, punct, or space keyword can be specified. Characters defined by the upper and lower keywords are automatically included in this character class.

digit Defines numeric digit characters. Only the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 can be specified.

space Defines white space characters. No character defined by the upper, lower, alpha, digit, graph, or xdigit keyword can be specified. At a minimum, the <space>, <form-feed>, <newline>, <carriage return>, <tab>, <vertical-tab> characters, and any characters defined by the blank keyword, are automatically included.

cntrl Defines control characters. No character defined by the upper, lower, alpha, digit, punct, graph, print, or xdigit keywords can be specified.

punct Defines punctuation characters. A character defined as the <space> character and characters defined by the upper, lower, alpha, digit, cntrl, or xdigit keyword cannot be specified.

graph Defines printable characters, excluding the <space> character. If this keyword is not specified, characters defined by the upper, lower, alpha, digit, xdigit, and punct keywords are automatically included in this character class. No character defined by the cntrl keyword can be specified.

print Defines printable characters, including the <space> character. If this keyword is not specified, the <space> character and characters defined by the upper, lower, alpha, digit, xdigit, and punct keywords are automatically included in this character class. No character defined by the cntrl keyword can be specified.

xdigit Defines hexadecimal digit characters. Only the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 and the letters A, B, C, D, E, F, a, b, c, d, e, and f can be specified. If not specified, the xdigit class defaults to the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 and the letters A, B, C, D, E, F, a, b, c, d, e, and f.

blank Defines blank characters. If this keyword is not specified, the <space> and <horizontal-tab> characters are included in this character class.

toupper
Defines the mapping of lowercase characters to uppercase characters. Operands for this keyword consist of semicolon-separated character pairs. Each character pair is enclosed in ( ) (parentheses) and separated from the next pair by a , (comma). The first character in each pair is considered lowercase; the second character is considered uppercase. Only characters defined by the lower and upper keywords can be specified.

tolower
Defines the mapping of uppercase characters to lowercase characters. Operands for this keyword consist of semicolon-separated character pairs. Each character pair is enclosed in ( ) (parentheses) and separated from the next pair by a , (comma). The first character in each pair is considered uppercase; the second character is considered lowercase. Only characters defined by the lower and upper keywords can be specified.

Note: The **tolower** keyword is optional. If this keyword is not specified, the mapping defaults to the reverse mapping of the **toupper** keyword, if specified. If the **toupper** keyword is not specified, the mapping defaults to the **C** locale.

Example:

The following is an example of a LC_CTYPE category in a locale definition source file:

```
LC_CTYPE

#"alpha" is by default "upper" and "lower"
```

```
#"print" is by default "alpha", "digit", "punct", and the space character
#"graph" is by default "alnum" and "punct"
#"tolower" is by default the reverse mapping of "toupper"
#
upper    <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>;<Z>
#
lower <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>;<z>
#
digit <zero>;<one>;<two>;<three>;<four>;<five>;<six>;\
<seven>;<eight>;<nine>
#
space <tab>;<newline>;<vertical-tab>;<form-feed>;\
<carriage-return>;<space>
#
cntrl <alert>;<backspace>;<tab>;<newline>;<vertical-tab>;\
<form-feed>;<carriage-return>;<NUL>;<SOH>;<STX>;\
<ETX>;<EOT>;<ENQ>;<ACK>;<SO>;<SI>;<DLE>;<DC1>;<DC2>;\
<DC3>;<DC4>;<NAK>;<SYN>;<ETB>;<CAN>;<EM>;<SUB>;\
<ESC>;<IS4>;<IS3>;<IS2>;<IS1>;<DEL>
#
punct <exclamation-mark>;<quotation-mark>;<number-sign>;\
<dollar-sign>;<percent-sign>;<ampersand>;<asterisk>;\
<apostrophe>;<left-parenthesis>;<right-parenthesis>;\
<plus-sign>;<comma>;<hyphen>;<period>;<slash>;\
<colon>;<semicolon>;<less-than-sign>;<equals-sign>;\
<greater-than-sign>;<question-mark>;<commercial-at>;\
<left-square-bracket>;<backslash>;<circumflex>;\
<right-square-bracket>;<underline>;<grave-accent>;\
<left-curly-bracket>;<vertical-line>;<tilde>;\
<right-curly-bracket>
#
xdigit <zero>;<one>;<two>;<three>;<four>;<five>;<six>;\
<seven>;<eight>;<nine>;<A>;<B>;<C>;<D>;<E>;<F>;\
<a>;<b>;<c>;<d>;<e>;<f>
#
blank <space>;<tab>
#
toupper (<a>,<A>);(<b>,<B>);(<c>,<C>);(<d>,<D>);(<e>,<E>);\
(<f>,<F>);(<g>,<G>);(<h>,<H>);(<i>,<I>);(<j>,<J>);\
(<k>,<K>);(<l>,<L>);(<m>,<M>);(<n>,<N>);(<o>,<O>);\
(<p>,<P>);(<q>,<Q>);(<r>,<R>);(<s>,<S>);(<t>,<T>);\
(<u>,<U>);(<v>,<V>);(<w>,<W>);(<x>,<X>);(<y>,<Y>);\
(<z>,<Z>)
#
END LC_CTYPE
```

LC_MESSAGES category: The LC_MESSAGES category of a locale definition source file defines the format for affirmative and negative system responses. This category begins with an LC_MESSAGES category header and ends with an END LC_MESSAGES category trailer.

All operands for the LC_MESSAGES category are defined as strings or *extended regular expressions* enclosed by double-quotation marks (″″).

Note: For additional information, see the “Extended regular expressions” on page 138 topic, below. These operands are separated from the keyword they define by one or more blanks. Two adjacent double-quotation marks (″″) indicate an undefined value. The following keywords are recognized in the LC_MESSAGES category:

yesexpr
: Specifies an extended regular expression that describes the acceptable affirmative response to a question expecting an affirmative or negative response.

noexpr
Specifies an extended regular expression that describes the acceptable negative response to a question expecting an affirmative or negative response.

yesstr A fixed string of acceptable affirmative response.

nostr A fixed string of acceptable negative response.

Extended regular expressions: The following special characters are used to form extended regular expressions:

Character	Function
+	Specifies that a string matches if one or more occurrences of the character or extended regular expression that precedes the + (plus) are within the string.
?	Specifies that a string matches if zero or one occurrences of the character or extended regular expression that precedes the ? (question mark) are within the string.
|	Specifies that a string matches if either of the strings separated by the | (vertical line) are within the string.
( )	Groups strings together in regular expressions.
{m}	Specifies that a string matches if exactly m occurrences of the pattern are within the string.
{m,}	Specifies that a string matches if at least m occurrences of the pattern are within the string.
{m, n}	Specifies that a string matches if between m and n, inclusive, occurrences of the pattern are within the string (where m <= n).
[String]	Signifies that the regular expression matches any characters specified by the string variable within the square brackets.
[^ String]	A ^ (caret) within the [ ] (square brackets) and at the beginning of the specified string indicates that the regular expression does not match any characters within the square brackets.
^	Signifies the beginning of a field or record.
$	Signifies the end of a field or record.
. (period)	Signifies any one character except the terminal new-line character at the end of a space.
*** (asterisk)**	Signifies zero or more of any characters.
\ (backslash)	The escape character. When preceding any of the characters that have special meaning in extended regular expressions, the escape character removes any special meaning for the character.

Character class expressions may also be specified in the extended regular expression. The following character class expressions are supported in all locales:

```
[:alnum:]
[:alpha:]
[:blank:]
[:cntrl:]
[:digit:]
[:graph:]
[:lower:]
[:print:]
[:punct:]
[:space:]
[:upper:]
[:xdigit:]
```

Example:

The following is an example of a LC_MESSAGES category in a locale definition source file:

```
LC_MESSAGES
#
yesexpr  "[yY]"
noexpr   "[nN]"
yesstr   "yes"
nostr    "no"
#
END LC_MESSAGES
```

LC_MONETARY category: The LC_MONETARY category of a locale definition source file defines rules and symbols for formatting monetary numeric information. This category begins with an LC_MONETARY category header and ends with an END LC_MONETARY category trailer.

All operands for the LC_MONETARY category keywords are defined as string or integer values. String values are bounded by double-quotation marks (“”). All values are separated from the keyword they define by one or more spaces. Two adjacent double-quotation marks indicate an undefined string value. A -1 indicates an undefined integer value. The following keywords are recognized in the LC_MONETARY category:

int_curr_symbol
Specifies the string used for the international currency symbol. The operand for the int_curr_symbol keyword is a four-character string. The first three characters contain the alphabetic international-currency symbol. The fourth character specifies a character separator between the international currency symbol and a monetary quantity.

currency_symbol
Specifies the string used for the local currency symbol.

mon_decimal_point
Specifies the string used for the decimal delimiter used to format monetary quantities.

mon_thousands_sep
Specifies the string used for grouping digits to the left of the decimal delimiter in formatted monetary quantities.

mon_grouping
Defines the size of each group of digits in formatted monetary quantities. The operand for the mon_grouping keyword consists of a sequence of semicolon-separated integers. Each integer specifies the number of digits in a group. The initial integer defines the size of the group immediately to the left of the decimal delimiter. The following integers define succeeding groups to the left of the previous group. If the last digit is not -1, subsequent grouping is performed using the previous digit. If the last digit is -1, grouping is only performed for the number of groups specified.

The following is an example of the interpretation of the mon_grouping keyword. Assuming the value to be formatted is 123456789 and the operand for the mon_thousands_sep keyword is comma (,), the following results occur:

mon_grouping Value	**Formatted Value**
3;-1	123456,789
3	123,456,789
3;2	12,34,56,789
3;2;-1	1234,56,789

positive_sign
Specifies the string used to indicate a nonnegative-valued formatted monetary quantity.

negative_sign
Specifies the string used to indicate a negative-valued formatted monetary quantity.

int_frac_digits
Specifies an integer value representing the number of fractional digits (those after the decimal delimiter) to be displayed in a formatted monetary quantity using the int_curr_symbol value.

frac_digits
Specifies an integer value representing the number of fractional digits (those after the decimal delimiter) to be displayed in a formatted monetary quantity using the currency_symbol value.

p_cs_precedes
Specifies an integer value indicating whether the int_curr_symbol or currency_symbol string precedes or follows the value for a non-negative formatted monetary quantity. The following integer values are recognized:

0 Indicates that the currency symbol follows the monetary quantity.

1 Indicates that the currency symbol precedes the monetary quantity.

p_sep_by_space
Specifies an integer value indicating whether the int_curr_symbol or currency_symbol string is separated by a space from a non-negative formatted monetary quantity. The following integer values are recognized:

0 Indicates that no space separates the currency symbol from the monetary quantity.

1 Indicates that a space separates the currency symbol from the monetary quantity.

2 Indicates that a space separates the currency symbol and the positive_sign string, if adjacent.

n_cs_precedes
Specifies an integer value indicating whether the int_curr_symbol or currency_symbol string precedes or follows the value for a negative formatted monetary quantity. The following integer values are recognized:

0 Indicates that the currency symbol follows the monetary quantity.

1 Indicates that the currency symbol precedes the monetary quantity.

n_sep_by_space
Specifies an integer value indicating whether the int_curr_symbol or currency_symbol string is separated by a space from a negative formatted monetary quantity. The following integer values are recognized:

0 Indicates that no space separates the currency symbol from the monetary quantity.

1 Indicates that a space separates the currency symbol from the monetary quantity.

2 Indicates that a space separates the currency symbol and the **negative_sign** string, if adjacent.

p_sign_posn
Specifies an integer value indicating the positioning of the positive_sign string for a non-negative formatted monetary quantity. The following integer values are recognized:

0 Indicates that parenthesis enclose both the monetary quantity and the int_curr_symbol or currency_symbol string.

1 Indicates that the positive_sign string precedes the quantity and the int_curr_symbol or currency_symbol string.

2 Indicates that the positive_sign string follows the quantity and the int_curr_symbol or currency_symbol string.

3 Indicates that the positive_sign string immediately precedes the int_curr_symbol or currency_symbol string.

4 Indicates that the positive_sign string immediately follows the int_curr_symbol or currency_symbol string.

n_sign_posn
Specifies an integer value indicating the positioning of the negative_sign string for a negative formatted monetary quantity. The following integer values are recognized:

0 Indicates that parenthesis enclose both the monetary quantity and the int_curr_symbol or currency_symbol string.

1 Indicates that the negative_sign string precedes the quantity and the int_curr_symbol or currency_symbol string.

2 Indicates that the negative_sign string follows the quantity and the int_curr_symbol or currency_symbol string.

3 Indicates that the negative_sign string immediately precedes the int_curr_symbol or currency_symbol string.

4 Indicates that the negative_sign string immediately follows the int_curr_symbol or currency_symbol string.

Examples:

The following is an example of the LC_MONETARY category listed in a locale definition source file:

```
LC_MONETARY
#
int_curr_symbol    "<U><S><D>"
currency_symbol    "<dollar-sign>"
mon_decimal_point  "<period>"
mon_thousands_sep  "<comma>"
mon_grouping       3;-1
positive_sign      "<plus-sign>"
negative_sign      "<hyphen>"
int_frac_digits    2
frac_digits        2
p_cs_precedes      1
p_sep_by_space     2
n_cs_precedes      1
n_sep_by_space     2
p_sign_posn        3
n_sign_posn        3
#
END LC_MONETARY
```

See Example: Producing unique monetary formats for another example relating to monetary formats.

LC_NUMERIC category: Defines rules and symbols for formatting non-monetary numeric information.

The LC_NUMERIC category of a locale definition source file defines rules and symbols for formatting non-monetary numeric information. This category begins with an LC_NUMERIC category header and terminates with an END LC_NUMERIC category trailer.

All operands for the LC_NUMERIC category keywords are defined as string or integer values. String values are bounded by double-quotation marks (″″). All values are separated from the keyword they define by one or more spaces. Two adjacent double-quotation marks indicate an undefined string value. A -1 indicates an undefined integer value. The following keywords are recognized in the LC_NUMERIC category:

decimal_point
Specifies a string containing the decimal delimiter character used to format numeric, non-monetary quantities.

thousands_sep
Specifies the string separator used for grouping digits to the left of the decimal delimiter in formatted numeric, non-monetary quantities.

grouping
Defines the size of each group of digits in formatted monetary quantities. The operand for the grouping keyword consists of a sequence of semicolon-separated integers. Each integer specifies the number of digits in a group. The initial integer defines the size of the group immediately to the left of the decimal delimiter. The following integers define succeeding groups to the left of the previous group. Grouping is performed for each integer specified for the grouping keyword. If the last digit is not -1, subsequent grouping is performed using the previous digit. If the last digit is -1, grouping is only performed for the number of groups specified.

The following is an example of the interpretation of the grouping statement. Assuming the value to be formatted is 123456789 and the operand for the thousands_sep keyword is comma (,), the following results occur:

Grouping value	Formatted value
3	123,456,789
3;-1	123456,789
3;2	12,34,56,789
3;2;-1	1234,56,789

Example:

Following is an example of a LC_NUMERIC category in a locale definition source file:

```
LC_NUMERIC
#
decimal_point "<period>"
thousands_sep "<comma>"
grouping      3;-1
#
END LC_NUMERIC
```

LC_TIME category: The LC_TIME category of a locale definition source file defines rules and symbols for formatting time and date information. This category begins with an LC_TIME category header and terminates with an END LC_TIME category trailer.

All operands for the LC_TIME category keywords are defined as string or integer values. String values are bounded by double-quotation marks (″″). All values are separated from the keyword they define by one or more spaces. Two adjacent double-quotation marks indicate an undefined string value. A -1 indicates an

undefined integer value. Field descriptors are used by commands and subroutines that query the LC_TIME category to represent elements of time and date formats. The following keywords are recognized in the LC_TIME category:

abday Defines the abbreviated weekday names corresponding to the %a field descriptor. Recognized values consist of seven semicolon-separated strings. The first string corresponds to the abbreviated name for the first day of the week (Sun), the second to the abbreviated name for the second day of the week, and so on.

day Defines the full spelling of the weekday names corresponding to the %A field descriptor. Recognized values consist of seven semicolon-separated strings. The first string corresponds to the full spelling of the name of the first day of the week (Sunday), the second to the name of the second day of the week, and so on.

abmon
Defines the abbreviated month names corresponding to the %b field descriptor. Recognized values consist of 12 semicolon-separated strings. The first string corresponds to the abbreviated name for the first month of the year (Jan), the second to the abbreviated name for the second month of the year, and so on.

mon Defines the full spelling of the month names corresponding to the %B field descriptor. Recognized values consist of 12 semicolon-separated strings. The first string corresponds to the full spelling of the name for the first month of the year (January), the second to the full spelling of the name for the second month of the year, and so on.

d_t_fmt
Defines the string used for the standard date and time format corresponding to the %c field descriptor. The string can contain any combination of characters, field descriptors, or escape sequences. See Escape Sequences (see page 145) for additional information.

d_fmt Defines the string used for the standard date format corresponding to the %x field descriptor. The string can contain any combination of characters, field descriptors, or escape sequences. Following is an example of how the d_fmt keyword can be constructed:

%D The %D indicates a %m/%d/%y date format. If you are using this format and have chosen to set the job attribute from the locale, then a '/' is extracted for the DATSEP job attribute. *MDY is extracted for the DATFMT job attribute.

%j The %j indicates a Julian date format. If you are using this format and have chosen to set the job attribute from the locale, then no DATSEP job is extracted. However, *JUL is extracted for the DATFMT job attribute.

%d-%m-%y
If you are using this format and have chosen to set the job attribute from the locale, then the compiler extracts - for the DATSEP job attribute and *DMY for the DATFMT job attribute.

%y.%m.%d
If you are using this format and have chosen to set the job attribute from the locale, then the compiler extracts . for the DATSEP job attribute and *YMD for the DATFMT job attribute.

%m/%d/%Y
If you are using this format and have chosen to set the job attribute from the locale, then the compiler extracts / for the DATSEP job attribute. No DATFMT job attribute is extracted.

Note: If the locale is to contain a valid OS/400 date format and date separator, then the d_fmt value must be defined such that it contains valid OS/400 date format and date separators. For example, if the value was specified as: *%m/%d/%y*, then *MDY would be extracted for the OS/400 date format and a / would be extracted for the OS/400 date format. A warning is issued by the CRTLOCALE command if an OS/400 date format or date separator cannot be extracted.

t_fmt Defines the string used for the standard time format corresponding to the %X field descriptor. The string can contain any combination of characters, field descriptors, or escape sequences. Following is an example of how the t_fmt keyword can be constructed:

%H:%M:%S
The compiler extracts a : (colon) for the TIMSEP job attribute.

%H.%M.%S
The compiler extracts a . (period) for the TIMSEP job attribute.

%H %M %S
The compiler extracts a blank space for the TIMSEP job attribute.

%H,%M,%S
The compiler extracts a , (comma) for the TIMSEP job attribute.

%T %T implies a %H:%M:%S (hours, minutes, seconds) time format with a : (colon) as the TIMSEP job attribute.

%H&%M&%S;
A valid TIMSEP job attribute could not be determined.

Note: If the locale is to contain a valid OS/400 time separator, then the t_fmt value must be defined such that it contains a valid OS/400 time separator. For example, if the value was specified as: *%H:%M:%S*, then a : (colon) would be extracted for the OS/400 date format. A warning is issued by the CRTLOCALE command if an OS/400 time separator cannot be extracted.

am_pm
Defines the strings used to represent *ante meridiem* (before noon) and *post meridiem* (after noon) corresponding to the %p field descriptor. Recognized values consist of two strings separated by a ; (semicolon). The first string corresponds to the *ante meridiem* designation, the last string to the *post meridiem* designation.

t_fmt_ampm
Defines the string used for the standard 12-hour time format that includes an am_pm value (%p field descriptor). This statement corresponds to the %r field descriptor. The string can contain any combination of characters and field descriptors.

era Defines how the years are counted and displayed for each era in a locale, corresponding to the %E field descriptor modifier. For each era, there must be one string in the following format:

direction:offset:start_date:end_date:era_name:era_format

The variables for the era-string format are defined as follows:

direction
Specifies a - (minus sign) or + (plus sign) character. The plus character indicates that years count in the positive direction when moving from the start date to the end date. The minus character indicates that years count in the negative direction when moving from the start date to the end date.

offset Specifies a number representing the first year of the era.

start_date
Specifies the starting date of the era in the *yyyy/mm/dd* format, where *yyyy, mm,* and *dd* are the year, month, and day, respectively. Years prior to the year AD 1 are represented as negative numbers. For example, an era beginning March 5th in the year 100 BC would be represented as -100/03/05.

end_date
Specifies the ending date of the era in the same form used for the *start_date* variable or one of the two special values -* or +*. A -* value indicates that the ending date of the era extends backward to the beginning of time. A +* value indicates that the ending date of the era extends forward to the end of time. Therefore, the ending date can be chronologically before or after the starting date of the era. For example, the strings for the Christian eras AD and BC would be entered as follows:

```
+:0:0000/01/01:+*:AD:%o %N
+:1:-0001/12/31:-*:BC:%o %N
```

era_name
Specifies a string representing the name of the era that is substituted for the %EC field descriptor.

era_format
Specifies a string for formatting the %EY field descriptor.

An **era** value consists of one string for each era. If more than one era is specified, each era string is separated by a ; (semicolon).

era_d_fmt
Defines the string used to represent the date in alternate-era format corresponding to the %Ex field descriptor. The string can contain any combination of characters and field descriptors.

era_t_fmt
Defines the string used to represent the time in alternate-era format corresponding to the %EX field descriptor. The string can contain any combination of characters and field descriptors.

era_d_t_fmt
Defines the string used to represent the date and time in alternate-era format corresponding to the %Ec field descriptor. The string can contain any combination of characters and field descriptors.

alt_digits
Defines alternate strings for digits corresponding to the %O field descriptor. Recognized values consist of a group of strings separated by ; (semicolons). The first string represents the alternate string for zero, the second string represents the alternate string for one, and so on. A maximum of 100 alternate strings can be specified.

Escape sequences

The following are escape sequences allowed for the d_t_fmt, d_fmt, and t_fmt keyword values:

>

****	Represents the backslash character.
\a	Represents the alert character.
\b	Represents the backspace character.
\f	Represents the form-feed character.
\n	Represents the newline character.
\r	Represents the carriage-return character.
\t	Represents the tab character.
\v	Represents the vertical-tab character.

Example:

The following is an example of a LC_TIME category in a locale definition source file:

```
LC_TIME
#
#Abbreviated weekday names (%a)
abday   "<S><u><n>";"<M><o><n>";"<T><u><e>";"<W><e><d>";\
         "<T><h><u>";"<F><r><i>";"<S><a><t>"
#
#Full weekday names (%A)
day "<S><u><n><d><a><y>";"<M><o><n><d><a><y>";\
"<T><u><e><s><d><a><y>";"<W><e><d><n><e><s><d><a><y>";\
"<T><h><u><r><s><d><a><y>";"<F><r><i><d><a><y>";\
"<S><a><t><u><r><d><a><y>"
#
#Abbreviated month names (%b)
abmon   "<J><a><n>";"<F><e><b>";"<M><a><r>";"<A><p><r>";\
         "<M><a><y>";"<J><u><n>";"<J><u><l>";"<A><u><g>";\
         "<S><e><p>";"<O><c><t>";"<N><o><v>";"<D><e><c>"
#
#Full month names (%B)
mon  "<J><a><n><u><a><r><y>";"<F><e><b><r><u><a><r><y>";\
"<M><a><r><c><h>";"<A><p><r><i><l>";"<M><a><y>";\
"<J><u><n><e>";"<J><u><l><y>";"<A><u><g><u><s><t>";\
"<S><e><p><t><e><m><b><e><r>";"<O><c><t><o><b><e><r>";\
"<N><o><v><e><m><b><e><r>";"<D><e><c><e><m><b><e><r>"
#
#Date and time format (%c)
d_t_fmt "%a %b %d %H:%M:%S %Y"
#
#Date format (%x)
d_fmt       "%m/%d/%y"
#
#Time format (%X)
t_fmt       "%H:%M:%S"
#
#Equivalent of AM/PM (%p)
am_pm       "<A><M>";"<P><M>"
#
#12-hour time format (%r)
t_fmt_ampm  "%I:%M:%S %p"
#
era      "+:0:0000/01/01:+*:AD:%EC";\
"+:1:-0001/12/31:-*:BC:%Ey";
era_d_fmt  ""
alt_digits    "<0><t><h>";"<1><s><t>";"<2><n><d>";"<3><r><d>";\
"<4><t><h>";"<5><t><h>";"<6><t><h>";"<7><t><h>";\
"<8><t><h>";"<9><t><h>";"<1><0><t><h>"
#
END LC_TIME
```

LC_TOD category: The LC_TOD category defines the rules used to define the start and end time of daylight savings time, the difference between local time and Greenwich Mean time, the time zone name, and the daylight savings time name. This category is an IBM extension and must appear after all other category definitions in the source file.

All the operands for the LC_TOD category are defined as string or integer values. String values are bounded by double-quotation marks (″″). All values are separated from the keyword they define by one or more spaces. Two adjacent double-quotation marks indicate an undefined string value. A 0 (zero) indicates an undefined integer value. The following keywords are recognized in the LC_TOD category.

tzdiff Specifies an integer value representing the time zone difference in minutes. It is the difference between the local time and Greenwich mean time.

tname Specifies the string used for the time zone name.

dstname
Specifies the string used for the daylight savings time name.

dststart

Specifies a set of four integers representing the start date for the daylight savings time. The operand for the dststart keyword consists of a sequence of four comma-separated integers in the following format:

month,week,day,time

The variables for the dststart format are defined as:

month

Specifies an integer value representing the month of the year when Daylight Savings Time (DST) takes effect. This value ranges from 1 to 12, with 1 corresponding to January, and 12 corresponding to December.

week Specifies an integer value representing the week of the month when DST takes effect. This value ranges from -4 to 4, with -4 corresponding to the fourth week of the month counting from the end of the month and 4 corresponding to the fourth week of the month counting from the beginning of the month.

day Specifies an integer value representing the day of the month when DST takes effect or if the **week** keyword is not 0 (zero), then this is the day of the week when DST takes effect. This value ranges from 1 to the last day of the month or 1 to the last day of the week.

time Specifies an integer value representing the number of seconds after 12 midnight, local standard time, when DST takes effect. This value ranges from 0 to 86399.

dstend

Specifies a set of four integers representing the end date for the daylight savings time. The operand for the **dstend** keyword consists of a sequence of four comma-separated integers in the following format:

month,week,day,time

The variables for the **dstend** format are defined as:

month

Specifies an integer value representing the month of the year when Daylight Savings Time (DST) ends. This value ranges from 1 to 12, with 1 corresponding to January, and 12 corresponding to December.

week Specifies an integer value representing the week of the month when DST ends. This value ranges from -4 to 4, with -4 corresponding to the fourth week of the month counting from the end of the month and 4 corresponding to the fourth week of the month counting from the beginning of the month.

day Specifies an integer value representing the day of the month when DST ends or if the week keyword is not 0 (zero), then this is the day of the week when DST ends. This value ranges from 1 to the last day of the month or 1 to the last day of the week.

time Specifies an integer value representing the number of seconds after 12 midnight, local standard time, when DST takes effect. This value ranges from 0 to 86399.

dstshift

Specifies an integer value representing the daylight savings time shift in seconds.

Example:

The following is an example of a LC_TOD category in a locale definition source file:

```
LC_TOD
#
tzdiff    -360
tname     "<C><e><n><t><r><a><l>"
dstname   "<C><D><T>"

#Set daylight savings time to start on 3rd week of October at
#midnight on Saturday.
dststart  10,3,6,0

#Set daylight savings time to end on April 23, at midnight.

dstend    4,0,23,0
dstshift  3600
#
END LC_TOD
```

Locale symbolic names

OS/400 supports locale symbolic names based on predefined names from the X/Open Standard portable character set. In addition, OS/400 supports a 5-character alphanumeric symbolic name for all characters, where:

- The first character of the symbolic name is a Latin capital letter U. This character identifies that the name is derived from the ISO/IEC 10646 Universal Coded Character Set.
- The second through fifth characters of the symbolic name represent the code point of the character in the ISO/IEC 10646 Universal Coded Character Set 2 Level 1. This portion of the symbolic name is assigned by code point for ease of creating and changing locales.

As an example, the question mark (?) character provides the following correlation between symbolic naming, UCS2-1 code point, and an IBM-assigned code point:

- The ? character is symbolically represented by `<question-mark>`
- It is at code point `U003F` in the ISO 10646 code page
- It is at code point `6F` in IBM code page 500.

Mapping of locale symbolic names provides a list of all symbolic names supported on OS/400. The table also provides the UCS2-1 (ISO 10646) code points, their corresponding IBM code page or code points, and a graphic representation of each character.

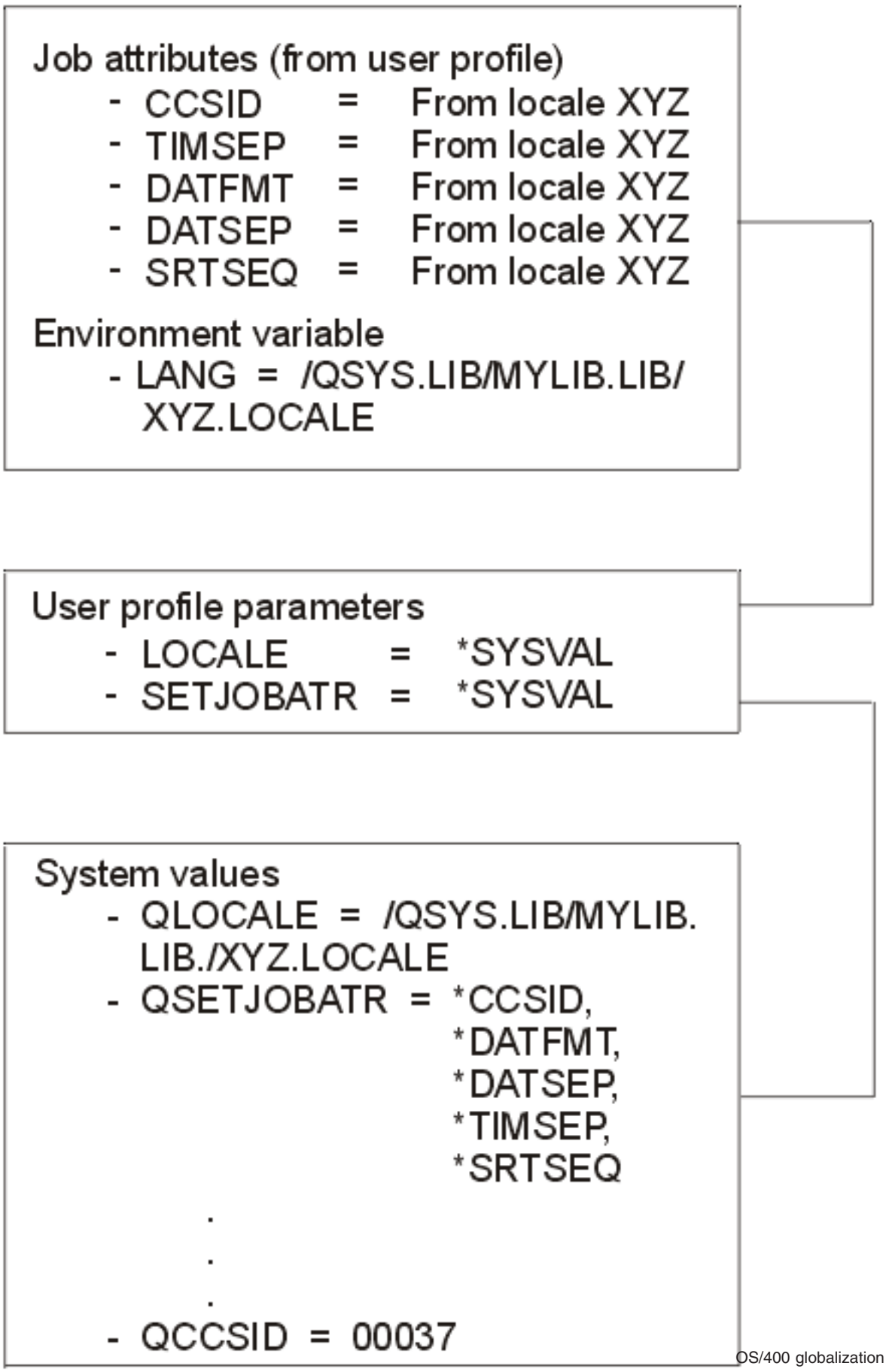
Examples: Locale programming

In addition to the following examples, Locale categories provides programming examples for each of the different locale categories:

- Example: How locales work
- Example: Creating locales
- Example: Producing unique monetary formats
- Example: Locales as part of a multilingual environment
- Example: POSIX locale
- Example: EN US locale

Example: How locales work: Following are two examples that focus on the LOCALE and SETJOBATR parameter values specified on the user profile.

The first example illustrates using locales to establish job attributes. The user profile parameters LOCALE and SETJOBATR have values of *SYSVAL. This means that the job attributes at job start up time come from the QLOCALE value based on the values in QSETJOBATR.


Job attributes (from user profile)
- CCSID = From locale XYZ
- TIMSEP = From locale XYZ
- DATFMT = From locale XYZ
- DATSEP = From locale XYZ
- SRTSEQ = From locale XYZ
Environment variable
- LANG = /QSYS.LIB/MYLIB.LIB/
XYZ.LOCALE
User profile parameters
- LOCALE = *SYSVAL
- SETJOBATR = *SYSVAL
System values
- QLOCALE = /QSYS.LIB/MYLIB.
LIB./XYZ.LOCALE
- QSETJOBATR = *CCSID,
*DATFMT,
*DATSEP,
*TIMSEP,
*SRTSEQ
.
.
.
- QCCSID = 00037

RBAGS522-0

If a job ran based on the information in the figure, the following would be true:

- The locale used would be XYZ.

  This is because the user profile parameter value for LOCALE was *SYSVAL. The *SYSVAL value is XYZ.
- The CCSID would be based on the value specified when locale XYZ was created.

  This value is specified when the LOCALE object is created using the CRTLOCALE command.
- The time separator would be derived from locale XYZ.

  This value is derived from the LC_TIME category specified in LOCALE XYZ.
- The date format separator would be derived from locale XYZ.

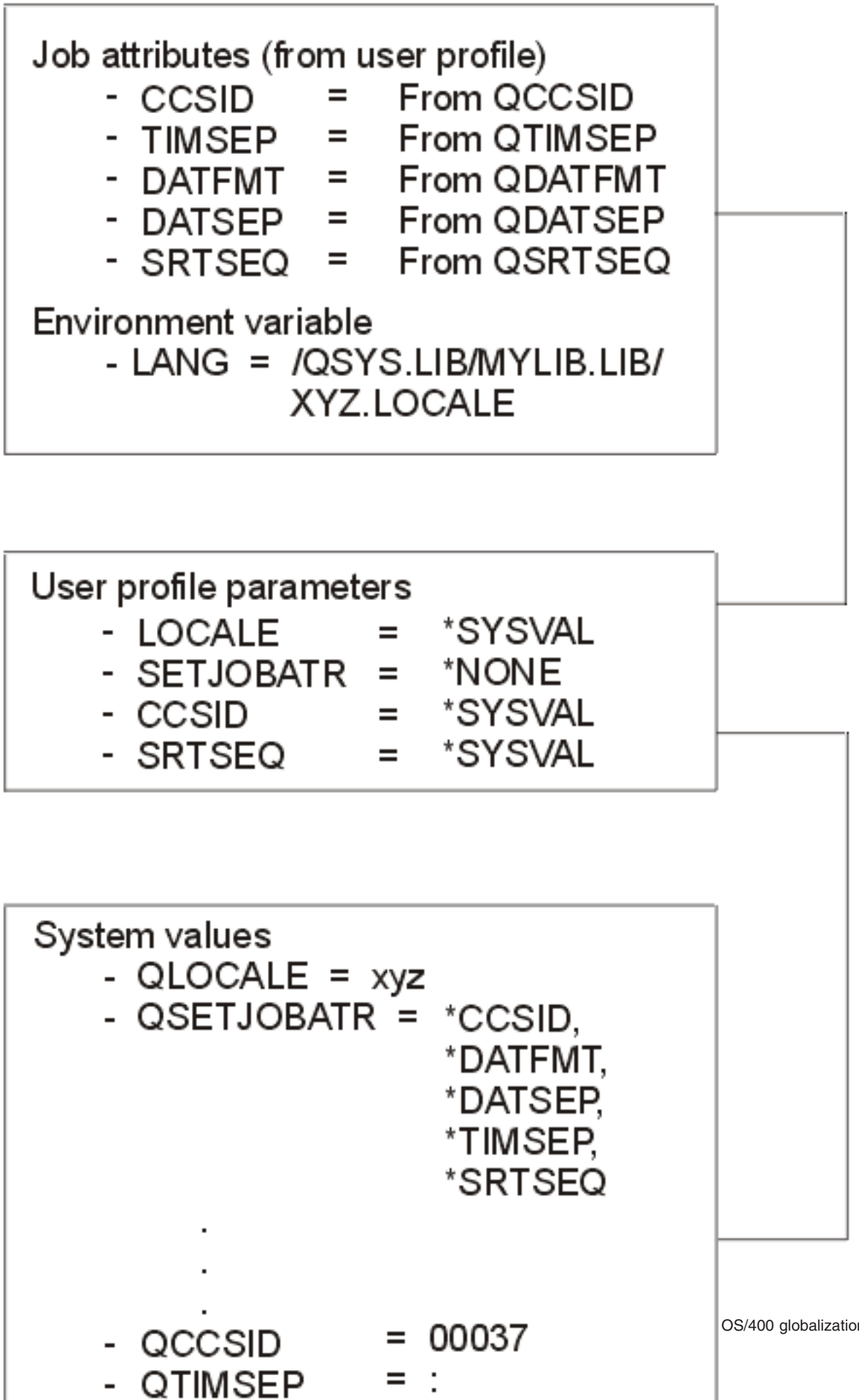
  This value is derived from the LC_TIME category specified in LOCALE XYZ.
- The data separator would be derived from locale XYZ.

  This value is derived from the LC_TIME category specified in LOCALE XYZ.
- The decimal format character would be derived from locale XYZ.

  This value is derived from the LC_NUMERIC category specified in LOCALE XYZ.

In the second example the user profile LOCALE parameter value is *SYSVAL and the SETJOBATR parameter values is *NONE. This means that the LOCALE value is determined by looking at the system value QLOCALE. When the SETJOBATR value is *NONE, job attributes are determined by the values in the user profile.

Remember, because the user profile SETJOBATR parameter was *NONE, the system's search resulted in using the values specified for QCCSID, QTIMSEP, QDATFMT, QSRTSEQ, and QDATSEP.


Job attributes (from user profile)
- CCSID = From QCCSID
- TIMSEP = From QTIMSEP
- DATFMT = From QDATFMT
- DATSEP = From QDATSEP
- SRTSEQ = From QSRTSEQ
Environment variable
- LANG = /QSYS.LIB/MYLIB.LIB/
XYZ.LOCALE
User profile parameters
- LOCALE = *SYSVAL
- SETJOBATR = *NONE
- CCSID = *SYSVAL
- SRTSEQ = *SYSVAL
System values
- QLOCALE = xyz
- QSETJOBATR = *CCSID,
*DATFMT,
*DATSEP,
*TIMSEP,
*SRTSEQ
.
.
.
- QCCSID = 00037
- QTIMSEP = :

If a job ran based on the information in this example, the following would be true:

- The locale used would be XYZ.

  This is because the user profile parameter value for LOCALE was *SYSVAL. The *SYSVAL value is XYZ.
- The CCSID is 00037.

  This is because the user profile SETJOBATR parameter value was *NONE. The system search ended with the value for QCCSID being used.
- The time separator is a colon (:).

  This is because the user profile SETJOBATR parameter value was *NONE. The system search ended with the value for QTIMSEP being used.
- The date format separator is a slash (/).

  This is because the user profile SETJOBATR parameter value was *NONE. The system search ended with the value for QDATSEP being used.
- The date format is month/day/year (MDY).

  This is because the user profile SETJOBATR parameter value was *NONE. The system search ended with the value for QDATFMT being used.
- The decimal format character is a period. Zero suppression is performed.

  This is because the user profile SETJOBATR parameter value was *NONE. The system search ended with the value for QDECFMT being used.

Example: Creating a locale: This example contains the steps necessary for creating a locale. The example also shows how to enable the locale. The steps are:

1. Create (or have) a library and a source physical file.
2. Copy an existing locale source file definition member (to a library and source physical file).
3. Edit the copied locale source file member if you need to customize any of the categories within the locale source.
4. Create the locale object.
5. Enable the locale object by using system values or parameters on the user profile.

Step 1. Create a library and source physical file

The library and source physical file are needed to store the locale source file member. See System-supplied locale source definition files for a list of the locale source file members that are shipped with OS/400.

1. Type CRTLIB and press the F4 (prompt) key.
2. Type localelib for the name of the library and press the Enter key.

There is now a library called localelib.

Next, create a source physical file.

1. Type CRTSRCPF and press the F4 (prompt) key.
2. Type localesrc for the file name and press the Enter key.

There is now a source physical file (localesrc) created in library localelib.

Step 2. Copy an existing locale source definition

IBM-supplied locale source definition file members are located in library QSYSLOCALE, source physical file QLOCALESRC. See System-supplied locale source definition files for a list of all IBM-supplied locale source files. In this example we will copy member EN_US, a locale for the English language.

1. Type CPYF and press the F4 (prompt) key.
2. Type the values shown on the following display.

```
+-------------------------------------------------------------------------------+
|Copy File (CPYF)                                                               |
|                                                                               |
|Type choices, press Enter.                                                     |
|                                                                               |
|From file  . . . . . .   QLOCALESRC    Name                                    |
|Library  . . . . . .     QSYSLOCALE  Name, *LIBL, *CURLIB                      |
|To file  . . . . . .     LOCALESRC_    Name, *PRINT                            |
|Library  . . . . . .     LOCALELIB_  Name, *LIBL, *CURLIB                      |
|From member  . . . . .   EN_US         Name, generic*, *FIRST, *ALL            |
|To member or label . .   EN_US_____    Name, *FIRST, *FROMMBR                  |
|Replace or add records   *ADD____      *NONE, *ADD, *REPLACE                   |
|Create file  . . . . .   *YES          *NO, *YES                               |
|Record format field mapping  . .   *MAP__        *NONE, *NOCHK, *CVTSRC        |
+-------------------------------------------------------------------------------+
```

   The values entered copy the EN_US member to the source physical file localesrc in library localelib.

Note: When you copy a file that is tagged with a CCSID, you need to use the FMTOPT(*MAP) parameter to ensure that the copied source is converted to the CCSID of the “to file”. The FMTOPT parameter can be seen by scrolling ahead.

Step 3. Edit the copied locale source definition

If you want to use the IBM-supplied locale as it is shipped, you do not need to change it. You can go to the next step, create the locale object. However, in this example we will edit the EN_US member to set the time-of-day keywords used in the LC_TOD category.

Note: The LC_TOD category is shipped with the keywords having no values. See the LC_TOD Category to view the LC_TOD source as shipped by IBM.

In this example, we are using Source Entry Utility (SEU) to edit the locale. You can use SEU or an equivalent editor.

1. Type STRSEU (Start Source Entry Utility) and press the F4 (prompt) key.
2. Type the source file name (localesrc), library name (localelib), and source member name (EN_US) as shown on the following display.

```
+-------------------------------------------------------------------------------+
|Start Source Entry Utility (STRSEU)                                            |
|                                                                               |
|Type choices, press Enter.                                                     |
|                                                                               |
|Source file  . . . . . .   localesrc     Name, *PRV                            |
|Library  . . . . . .       localelib_  Name, *LIBL, *CURLIB, *PRV              |
|Source member  . . . . .   EN_US_____    Name, *PRV, *SELECT                   |
+-------------------------------------------------------------------------------+
```

3. Press the Enter key. The following display appears:

```
+-------------------------------------------------------------------------------+
|COLUMNS . . . :    1  71          EDIT      LOCALELIB/LOCALESRC                |
|SEU==> F_LC_TOD      EN_US                                                     |
|FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7|
|*************** BEGINNING OF DATA ************************************        |
|5967.00 comment_char <percent-sign>                                            |
|5968.00 escape_char  <slash>                                                   |
|5969.00                                                                        |
```

```
5970.00 %                                                                      |
5971.00 % 5716SS1         (C) COPYRIGHT IBM CORP. 1991,1996                    |
5972.00 % ALL RIGHTS RESERVED.                                                 |
5973.00 % US GOVERNMENT USERS RESTRICTED RIGHTS -                              |
5974.00 % USE, DUPLICATION OR DISCLOSURE RESTRICTED                            |
5975.00 % BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.                          |
5976.00 %                                                                      |
5977.00 % LICENSED MATERIALS-PROPERTY OF IBM                                   |
5978.00 %                                                                      |
5979.00 % FILE NAME  :    EN_US                                                |
5980.00 %                                                                      |
5981.00 % COUNTRY/REGION: UNITED STATES                                        |
5982.00 %                                                                      |
                                                                               |
F3=EXIT   F4=PROMPT   F5=REFRESH   F9=RETRIEVE   F10=CURSOR   F11=TOGGLE       |
F16=REPEAT FIND       F17=REPEAT CHANGE          F24=MORE KEYS                 |
                                  (C) COPYRIGHT IBM CORP. 1981, 1996.          |
+------------------------------------------------------------------------------+
```

4. Use the SEU search function to locate LC_TOD. After the search completes, the display below appears.

   As you can see, all LC_TOD category keywords have values of 0 and no descriptive names declared for tname and dstname.

```
+------------------------------------------------------------------------------+
|COLUMNS . . . :    1  71          EDIT      LOCALELIB/LOCALESRC               |
|SEU==>       EN_US                                                            |
|FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 |
|6519.00                                                                       |
|6520.00 LC_TOD                                                                |
|6521.00                                                                       |
|6522.00 tzdiff   0                                                            |
|6523.00 tname    ""                                                           |
|6524.00 dstname  ""                                                           |
|6525.00 dststart 0,0,0,0                                                      |
|6526.00 dstend   0,0,0,0                                                      |
|6527.00 dstshift 0                                                            |
|6528.00                                                                       |
|6529.00 END LC_TOD                                                            |
|****************** END OF DATA ***************************************         |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
|                                                                              |
|F3=EXIT   F4=PROMPT   F5=REFRESH   F9=RETRIEVE   F10=CURSOR   F11=TOGGLE      |
|F16=REPEAT FIND       F17=REPEAT CHANGE          F24=MORE KEYS                |
|STRING LC_TOD FOUND.                                                          |
+------------------------------------------------------------------------------+
```

5. Type the following values for the LC_TOD keywords. If you need more detailed information about this category, see LC_TOD Category.

   tzdiff Time zone difference in minutes. Type -360. This is the difference in the number of minutes between Greenwich mean time and the central time zone of the United States.

   tname Specifies the string used for the time zone name. Type “<C><E><N><T><R><A><L>”.

   dstname
   The string used for the daylight savings time name. Type “<C><D><T>” to mean Central Daylight Time.

   dststart
   Four integers representing the start date for daylight savings time. Type 4,0,23,0. This string of integers means that daylight savings time starts the fourth month of the year, the twenty-third day of the month, and 0 seconds after midnight local standard time.

dstend

Four integers representing the end date for daylight savings time. Type 10,3,6,0. This string of integers means that daylight savings time ends the tenth month of the year, the beginning of the third week, the sixth day of the week, and 0 seconds after midnight local standard time.

dstshift

An integer value representing the daylight savings time shift in seconds. Type 3600.

The SEU edit screen should match the screen below.

6. Press the Enter key to make the changes to the EN_US locale member.

```
+------------------------------------------------------------------------------+
|COLUMNS . . . :    1  71          EDIT      LOCALELIB/LOCALESRC               |
|SEU==>       EN_US                                                           |
|FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 |
|6519.00                                                                      |
|6520.00 LC_TOD                                                               |
|6521.00                                                                      |
|6522.00 tzdiff   -360                                                        |
|6523.00 tname    "<C><E><N><T><R><A><L>"                                     |
|6524.00 dstname  "<C><D><T>"                                                 |
|6525.00 dststart 4,0,23,0                                                    |
|6526.00 dstend   10,3,6,0                                                    |
|6527.00 dstshift 3600                                                        |
|6528.00                                                                      |
|6529.00 END LC_TOD                                                           |
|****************** END OF DATA ***************************************       |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|F3=EXIT   F4=PROMPT   F5=REFRESH   F9=RETRIEVE   F10=CURSOR   F11=TOGGLE     |
|F16=REPEAT FIND       F17=REPEAT CHANGE          F24=MORE KEYS               |
|STRING LC_TOD FOUND.                                                         |
+------------------------------------------------------------------------------+
```

7. Save the member and exit SEU.

You have now copied the IBM-supplied locale source member and customized the LC_TOD category.

Step 4. Create the locale object

1. Type CRTLOCALE on any command line and press the F4 (prompt) key. The Create Locale (CRTLOCALE) screen appears.
2. Type the locale path name (path name includes the location of as well as the name of the locale source member) in the locale name field.
3. Next, type the source file path name (location and name of the source physical file and the member name) in the source file path name field.

   Notes:

   a. Make sure that the locale source file you are using has the same characters defined as does the CCSID you specify on the CRTLOCALE command. If they do not have the same characters available, unpredictable results can occur. For example, the EN_US source file has lower case letters defined. However, the lower case letters are not available in CCSID 290.

   b. When creating locale objects, it is recommended to make the CCSID part of the locale object name. For example, EN_US created with CCSID 37 could be named EN_US37.

```
+------------------------------------------------------------------------------+
|Create Locale (CRTLOCALE)                                                     |
|                                                                              |
|Type choices, press Enter.                                                    |
|                                                                              |
```

```
Locale name  . . . . . > '/qsys.lib/localelib.lib/en_us.locale'____
_____
Source file path name  . . . . . > '/qsys.lib/localelib.lib/localesrc.file/en
_us.mbr'
Coded character set ID . . . . . > 37_______    1-65533, *JOB
Generation severity level  . . .   10____      10, 20
Text 'description' . . . . . .   my version of locale EN_US - contains my c
hanges_
Bottom
F3=Exit   F4=Prompt   F5=Refresh   F10=Additional parameters   F12=Cancel
F13=How to use this display        F24=More keys
```

4. Press the Enter key to complete the creation of the locale object named EN_US in the library LOCALELIB

Step 5. Enable the locale object

Locales can be enabled system-wide using the QLOCALE system value or for individual users by changing their user profile. To enable system-wide, make EN_US the value for QLOCALE. In this example we will enable locale support for one user.

1. Type CHGUSRPRF and press the F4 (prompt) key.
2. Specify your userid and then press the Enter key.

   In the portion of the Change User Profile display shown below, the LOCALE parameter now has a value indicating that EN_US is the specified locale to be used by your userid.

```
+ for more values
Locale . . . . . .   QSYS.LIB/LOCALELIB.LIB/EN_US.LOCALE_____
_______
```

After your user profile has been changed, any jobs initiated by your userid have the EN_US locale associated with those jobs. The LANG environment variable is also initialized to the name of the locale.

Example: Producing unique monetary formats: A unique customized monetary format can be produced by changing the value of a single statement. For example, the following table shows the results of using all combinations of defined values for the p_cs_precedes, p_sep_by_space, and p_sign_posn statements:

p_cs_precedes value	p_sign_posn value	p_sep_by_space=2	p_sep_by_space=3	p_sep_by_space=4
p_cs_precedes = 1	p_sign_posn = 0	($1.25)	($ 1.25)	($1.25)
	p_sign_posn = 1	+ $1.25	+$ 1.25	+$1.25
	p_sign_posn = 2	$1.25 +	$ 1.25+	$1.25+
	p_sign_posn = 3	+ $1.25	+$ 1.25	+$1.25
	p_sign_posn = 4	$ +1.25	$+ 1.25	$+1.25

p_cs_precedes value	p_sign_posn value	p_sep_by_space=2	p_sep_by_space=3	p_sep_by_space=4
p_cs_precedes = 0	p_sign_posn = 0	(1.25 $)	(1.25 $)	(1.25$)
	p_sign_posn = 1	+1.25 $	+1.25 $	+1.25$
	p_sign_posn = 2	1.25$ +	1.25 $+	1.25$+
	p_sign_posn = 3	1.25+ $	1.25 +$	1.25+$
	p_sign_posn = 4	1.25$ +	1.25 $+	1.25$+

Example: Locales as part of a multilingual environment: OS/400, through the use of locales, user profiles, and subsystems, can provide a multilingual environment. Users of a system setup for multilingual environments work with their national language and all its cultural conventions (for example, the character used to separate hours, minutes, and seconds).

Assume the system used in this example has its primary language defined as English and the secondary national language versions (NLVs) for French and Spanish have been installed.

Follow the steps in this example to:

- Create the locales for English, French, and Spanish
- Create user profiles for users named: English, French, and Spanish
- Create separate subsystems for French and Spanish language users.

Step 1. Create locales

1. Type CRTLOCALE and press the Prompt key (F4).
2. Enter the following values for the fields listed below:
   - Locale name: qsys.lib/localelib.lib/en_us.locale
   - Source file path name: qsys.lib/qsyslocale.lib/qlocalesrc.file/en_us.mbr
   - Coded character set ID: 37
   - Generation severity level: 20
   - Text 'description': US English locale
3. Press Enter.

Repeat the CRTLOCALE command for the FRENCH and SPANISH locales, using the following values for the fields listed below.

For the French locale:

- Locale name: qsys.lib/localelib.lib/fr_fr.locale
- Source file path name: qsys.lib/qsyslocale.lib/qlocalesrc.file/fr_fr.mbr
- Coded character set ID: 297
- Generation severity level: 20
- Text 'description': French locale

For the Spanish locale:

- Locale name: qsys.lib/localelib.lib/es_es.locale
- Source file path name: qsys.lib/qsyslocale.lib/qlocalesrc.file/es_es.mbr
- Coded character set ID: 284
- Generation severity level: 20
- Text 'description': Spanish locale

You have created three locales (EN_US (English US), FR_FR (French), and ES_ES (Spanish). They are stored in library localelib.lib.

Step 2. Create the user profile

In this example three user profiles are created; each one will use one of locales we just created. The user profile names are: English, French, and Spanish.

1. Type CRTUSRPRF and prompt (F4).
2. Type ENGLISH for the User profile parameter value
3. Scroll forward until you see the Locale job attributes parameter and the Locale parameter.
4. Type:
   - /qsys.lib/localelib.lib/en_us.locale for the Locale parameter value.
   - Type + for the Locale job attributes parameter value and press Enter. Type:

     *CCSID

     *DATFMT

     *DATSEP

     *TIMSEP

     *SRTSEQ

     *DECFMT

   Note: At job start up the system finds the actual job attribute values defined in the locale object. The job attributes found in the locale override the values specified in the user profile fields for the CCSID and SRTSEQ parameters. They also override the Date Format, Date Separator, and Time Separator job attributes specified in any system value.
5. Press Enter. You have now created the user profile for a user named ENGLISH.

Repeat the CRTUSRPRF command for user IDs FRENCH and SPANISH. The next two displays provide the correct Locale parameter and Locale job attribute information for creating the user profiles for FRENCH and SPANISH.

```
+------------------------------------------------------------------------------+
|Locale job attributes  . . . . . > *CCSID        *SYSVAL, *NONE, *CCSID...    |
|> *DATFMT                                                                     |
|> *DATSEP                                                                     |
|> *TIMSEP                                                                     |
|> *SRTSEQ                                                                     |
|Locale . . . . . . . . . . . . . > '/qsys.lib/localelib.lib/fr_fr.locale'     |
+------------------------------------------------------------------------------+
```

```
+------------------------------------------------------------------------------+
|Locale job attributes  . . . . . > *CCSID        *SYSVAL, *NONE, *CCSID...    |
|> *DATFMT                                                                     |
|> *DATSEP                                                                     |
|> *TIMSEP                                                                     |
|> *SRTSEQ                                                                     |
|Locale . . . . . . . . . . . . . > '/qsys.lib/localelib.lib/es_es.locale'     |
+------------------------------------------------------------------------------+
```

Step 3. Creating subsystems for each national language version

Subsystems can be tailored to provide users an environment in which they see their own national language with data presented in the cultural format and conventions they are used to seeing.

Note: Since the primary language of the system is English, we do not need to create a subsystem for English.

1. Type CRTSBSD and prompt (F4).
2. Specify values for the following parameters to ensure that the subsystem is enabled for a specific national language (such as French and Spanish in our example).
   - Subsystem description

     This can be any name you choose.
   - Text 'description'

     The description can be anything you want it to be.
   - Sign-on display file and Library

     This often is QDSIGNON. The important information here is to know the name of the library where the national language version (French in this example) is stored.
   - Subsystem library

     Specifies a library that is entered ahead of other libraries in the library list of jobs started in this subsystem. This parameter allows you to use a secondary language library causing messages and displays to appear in your spoken language.

   Note: The correct values for Sign-on display file library and Subsystem library parameters are determined by adding QSYS to the national language version feature code. For example: the French national language library is named QSYS2928.

   See National language version feature codes for a listing of all supported language versions.

   The screen below shows the correct values to ensure that users of the FRENCH subsystem interact with the computer in the French language.

```
+------------------------------------------------------------------------------+
|Create Subsystem Description (CRTSBSD)                                        |
|                                                                              |
|Type choices, press Enter.                                                    |
|                                                                              |
|Subsystem description  . . . . . SBSD         > FRENCH                        |
|Library  . . . . . .                            *CURLIB                       |
+------------------------------------------------------------------------------+

+------------------------------------------------------------------------------+
|Text 'description' . . . . . . . TEXT         > 'Subsystem for French users'  |
|                                                                              |
|                                                                              |
|Additional Parameters                                                         |
|                                                                              |
|Sign-on display file . . . . . . SGNDSPF      > QDSIGNON                      |
|Library  . . . . . .                        >   QSYS2928                      |
|Subsystem library  . . . . . . . SYSLIBLE     > QSYS2928                      |
|More...                                                                       |
|F3=Exit   F4=Prompt   F5=Refresh   F12=Cancel   F13=How to use this display   |
|F24=More keys                                                                 |
|                                                                              |
|                                                                              |
+------------------------------------------------------------------------------+
```

3. Press Enter.

Step 4. More information about subsystems

Creation of a subsystem requires additional work such as:

- Setting subsystem attributes
- Adding workstation entires
- Adding job queue entries
- Adding communications entires (if your national language users are attached over communications lines)

- Adding autostart job entires if you want to use this feature
- Adding prestart job entires if you want to use this feature
- Creating a class
- Adding routing entries

How to perform the tasks in the list above is not described in this example. For more information about Subsystems, see the Work Management topic in the Information Center.

Example: POSIX locale: The POSIX (or C) locale follows. It is published in its entirety because:

- It provides a locale example with source provided for all categories.
- If you have not set a locale value in your C application program, the default POSIX locale is then used.

In either case, in the listing below, you are able to look at the locale categories and view the source.

```
     comment_char <percent-sign>
     escape_char  <slash>


%
% 5716SS1 (C) COPYRIGHT IBM CORP. 1991,1996
% ALL RIGHTS RESERVED.
% US GOVERNMENT USERS RESTRICTED RIGHTS -
% USE, DUPLICATION OR DISCLOSURE RESTRICTED
% BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
%
% LICENSED MATERIALS-PROPERTY OF IBM
%
% FILE NAME : POSIX
%
% COUNTRY/REGION: POSIX DEFAULT LOCALE
%
% LANGUAGES(S): NOT SPECIFIED
%
% DESCRIPTION: LOCALE SOURCE DEFINITION FILE.
%

LC_CTYPE

upper <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>;<Z>


lower <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>;<z>


space <tab>;<newline>;<vertical-tab>;<form-feed>;<carriage-return>;/
<space>

cntrl <NUL>;<SOH>;<STX>;<ETX>;<EOT>;<ENQ>;<ACK>;<alert>;<backspace>;/
<tab>;<newline>;<vertical-tab>;<form-feed>;<carriage-return>;/
<SO>;<SI>;<DLE>;<DC1>;<DC2>;<DC3>;<DC4>;<NAK>;<SYN>;<ETB>;/
<CAN>;<EM>;<SUB>;<ESC>;<IS4>;<IS3>;<IS2>;<IS1>;<DEL>


punct <exclamation-mark>;<quotation-mark>;<number-sign>;/
<dollar-sign>;<percent-sign>;<ampersand>;<apostrophe>;/
<left-parenthesis>;<right-parenthesis>;<asterisk>;<plus-sign>;/
<comma>;<hyphen>;<period>;<slash>;/
<colon>;<semicolon>;<less-than-sign>;/
<equals-sign>;<greater-than-sign>;<question-mark>;/
<commercial-at>;/
<left-square-bracket>;<backslash>;/
<right-square-bracket>;<circumflex>;/
```

```
<underscore>;<grave-accent>;/
<left-curly-bracket>;<vertical-line>;<right-curly-bracket>;/
<tilde>

digit <zero>;<one>;<two>;<three>;<four>;/
<five>;<six>;<seven>;<eight>;<nine>

xdigit <zero>;<one>;<two>;<three>;<four>;/
<five>;<six>;<seven>;<eight>;<nine>;/
<A>;<B>;<C>;<D>;<E>;<F>;/
<a>;<b>;<c>;<d>;<e>;<f>

blank <space>;/
<tab>

toupper (<a>,<A>);(<b>,<B>);(<c>,<C>);(<d>,<D>);(<e>,<E>);/
(<f>,<F>);(<g>,<G>);(<h>,<H>);(<i>,<I>);(<j>,<J>);/
(<k>,<K>);(<l>,<L>);(<m>,<M>);(<n>,<N>);(<o>,<O>);/
(<p>,<P>);(<q>,<Q>);(<r>,<R>);(<s>,<S>);(<t>,<T>);/
(<u>,<U>);(<v>,<V>);(<w>,<W>);(<x>,<X>);(<y>,<Y>);/
(<z>,<Z>)

tolower (<A>,<a>);(<B>,<b>);(<C>,<c>);(<D>,<d>);(<E>,<e>);/
(<F>,<f>);(<G>,<g>);(<H>,<h>);(<I>,<i>);(<J>,<j>);/
(<K>,<k>);(<L>,<l>);(<M>,<m>);(<N>,<n>);(<O>,<o>);/
(<P>,<p>);(<Q>,<q>);(<R>,<r>);(<S>,<s>);(<T>,<t>);/
(<U>,<u>);(<V>,<v>);(<W>,<w>);(<X>,<x>);(<Y>,<y>);/
(<Z>,<z>)

END LC_CTYPE

LC_COLLATE

order_start

<NUL>
<SOH>
<STX>
<ETX>
<EOT>
<ENQ>
<ACK>
<alert>
<backspace>
<tab>
<newline>
<vertical-tab>
<form-feed>
<carriage-return>
<SO>
<SI>
<DLE>
<DC1>
<DC2>
<DC3>
<DC4>
<NAK>
<SYN>
<ETB>
<CAN>
<EM>
<SUB>
<ESC>
<IS4>
<IS3>
<IS2>
<IS1>
```

```
<space>
<exclamation-mark>
<quotation-mark>
<number-sign>
<dollar-sign>
<percent-sign>
<ampersand>
<apostrophe>
<left-parenthesis>
<right-parenthesis>
<asterisk>
<plus-sign>
<comma>
<hyphen>
<period>
<slash>
<zero>
<one>
<two>
<three>
<four>
<five>
<six>
<seven>
<eight>
<nine>
<colon>
<semicolon>
<less-than-sign>
<equals-sign>
<greater-than-sign>
<question-mark>
<commercial-at>
<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>
<Z>
<left-square-bracket>
<backslash>
<right-square-bracket>
<circumflex>
<underscore>
<grave-accent>
<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>
<z>
<left-curly-bracket>
<vertical-line>
<right-curly-bracket>
<tilde>
<DEL>
UNDEFINED

order_end

END LC_COLLATE

LC_MONETARY

int_curr_symbol ""
currency_symbol ""
mon_decimal_point ""
mon_thousands_sep ""
mon_grouping -1
positive_sign ""
negative_sign ""
int_frac_digits -1
frac_digits -1
p_cs_precedes -1
p_sep_by_space -1
n_cs_precedes -1
n_sep_by_space -1
p_sign_posn -1
n_sign_posn -1

END LC_MONETARY

LC_NUMERIC

decimal_point "<period>" thousands_sep
"" grouping -1

END LC_NUMERIC

LC_TIME

abday "<S><u><n>";/
"<M><o><n>";/
"<T><u><e>";/
```

```
"<W><e><d>";/
"<T><h><u>";/
"<F><r><i>";/
"<S><a><t>"

day "<S><u><n><d><a><y>";/
"<M><o><n><d><a><y>";/
"<T><u><e><s><d><a><y>";/
"<W><e><d><n><e><s><d><a><y>";/
"<T><h><u><r><s><d><a><y>";/
"<F><r><i><d><a><y>";/
"<S><a><t><u><r><d><a><y>"


abmon "<J><a><n>";/
"<F><e><b>";/
"<M><a><r>";/
"<A><p><r>";/
"<M><a><y>";/
"<J><u><n>";/
"<J><u><l>";/
"<A><u><g>";/
"<S><e><p>";/
"<O><c><t>";/
"<N><o><v>";/
"<D><e><c>"


mon "<J><a><n><u><a><r><y>";/
"<F><e><b><r><u><a><r><y>";/
"<M><a><r><c><h>";/
"<A><p><r><i><l>";/
"<M><a><y>";/
"<J><u><n><e>";/
"<J><u><l><y>";/
"<A><u><g><u><s><t>";/
"<S><e><p><t><e><m><b><e><r>";/
"<O><c><t><o><b><e><r>";/
"<N><o><v><e><m><b><e><r>";/
"<D><e><c><e><m><b><e><r>"


d_t_fmt "%a %b %d %H:%M:%S %Z %Y"

d_fmt "%m//%d//%y"

t_fmt "%H:%M:%S"

am_pm "<A><M>";"<P><M>"

t_fmt_ampm "%I:%M:%S %p"

END LC_TIME

LC_MESSAGES

yesexpr "[yY][eE][sS]|[yY]"
noexpr "[nN][oO]|[nN]"
yesstr "yes"
nostr "no"

END LC_MESSAGES

LC_TOD

tzdiff 0
tname ""
```

```
dstname ""
dststart 0,0,0,0
dstend 0,0,0,0
dstshift 0

END LC_TOD
```

Example: EN_US locale: The EN-US locale follows. In the example below, you can look at the locale categories and view the source.

```
comment_char <percent-sign>
escape_char  <slash>

%
% 5716SS1         (C) COPYRIGHT IBM CORP. 1991,1996
% ALL RIGHTS RESERVED.
% US GOVERNMENT USERS RESTRICTED RIGHTS -
% USE, DUPLICATION OR DISCLOSURE RESTRICTED
% BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP.
%
% LICENSED MATERIALS-PROPERTY OF IBM
%
% FILE NAME  :    EN_US
%
% COUNTRY/REGION: UNITED STATES
%
% LANGUAGES(S):   ENGLISH
%
% DESCRIPTION:    LOCALE SOURCE DEFINITION FILE.
%


LC_CTYPE

upper    <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>;<Z>;/
<A-acute>;<A-grave>;<A-circumflex>;<A-diaresis>;/
<A-tilde>;<A-ring>;<AE>;<C-cedilla>;<Eth>;<E-acute>;/
<E-grave>;<E-circumflex>;<E-diaresis>;<I-acute>;/
<I-grave>;<I-circumflex>;<I-diaresis>;<N-tilde>;/
<O-acute>;<O-grave>;<O-circumflex>;<O-diaresis>;/
<O-tilde>;<O-slash>;<Thorn>;<U-acute>;<U-grave>;/
<U-circumflex>;<U-diaresis>;<Y-acute>

lower    <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>;<z>;/
<a-acute>;<a-grave>;<a-circumflex>;<a-diaresis>;/
<a-tilde>;<a-ring>;<ae>;<c-cedilla>;<eth>;<e-acute>;/
<e-grave>;<e-circumflex>;<e-diaresis>;<i-acute>;/
<i-grave>;<i-circumflex>;<i-diaresis>;<n-tilde>;/
<o-acute>;<o-grave>;<o-circumflex>;<o-diaresis>;/
<o-tilde>;<o-slash>;<s-sharp>;<thorn>;<u-acute>;/
<u-grave>;<u-circumflex>;<u-diaresis>;<y-acute>;/
<y-diaresis>

space    <tab>;<newline>;<vertical-tab>;<form-feed>;<carriage-return>;/
<space>

cntrl    <NUL>;<SOH>;<STX>;<ETX>;<EOT>;<ENQ>;<ACK>;<alert>;<backspace>;/
<tab>;<newline>;<vertical-tab>;<form-feed>;<carriage-return>;/
<SO>;<SI>;<DLE>;<DC1>;<DC2>;<DC3>;<DC4>;<NAK>;<SYN>;<ETB>;/
<CAN>;<EM>;<SUB>;<ESC>;<IS4>;<IS3>;<IS2>;<IS1>;<DEL>;/
<DS>;<SOS>;<FS>;<WUS>;<BYP>;<NL>;<RNL>;<POC>;<SA>;<SFE>;<SM>;/
<CSP>;<MFA>;<SPS>;<RPT>;<CU1>;<DCS>;<PU1>;<UBS>;<IR>;<PP>;/
<TRN>;<NBS>;<GE>;<SBS>;<IT>;<RFF>;<CU3>;<SEL>;<RES>;<PM>;<EO>

graph    <exclamation-mark>;<quotation-mark>;<number-sign>; /
```

```
<dollar-sign>;<percent-sign>;<ampersand>;<apostrophe>;     /
<left-parenthesis>;<right-parenthesis>;<asterisk>;<plus-sign>;/
<comma>;<hyphen-minus>;<period>;<slash>;/
<zero>;<one>;<two>;<three>;<four>;<five>;<six>;<seven>;/
<eight>;<nine>;<colon>;<semicolon>;<less-than-sign>;   /
<equals-sign>;<greater-than-sign>;<question-mark>;/
<commercial-at>;<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>;<Z>;/
<left-square-bracket>;<backslash>;/
<right-square-bracket>;<circumflex>;/
<underscore>;<grave-accent>;/
<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>;<z>;/
<left-brace>;<vertical-line>;<right-brace>;/
<tilde>;<C-cedilla>;<u-diaresis>;<e-acute>;<a-circumflex>;/
<a-diaresis>;<a-grave>;<a-ring>;<c-cedilla>;<e-circumflex>;/
<e-diaresis>;<e-grave>;<i-diaresis>;<i-circumflex>;/
<i-grave>;<A-diaresis>;<A-ring>;<E-acute>;<ae>;<AE>;/
<o-circumflex>;<o-diaresis>;<o-grave>;<u-circumflex>;/
<u-grave>;<y-diaresis>;<O-diaresis>;<U-diaresis>;<o-slash>;/
<sterling>;<O-slash>;<multiply>;<a-acute>;<i-acute>;/
<o-acute>;<u-acute>;<n-tilde>;<N-tilde>;<feminine>;/
<masculine>;<question-down>;<registered>;<not>;<one-half>;/
<one-quarter>;<exclamation-down>;<guillemot-left>;/
<guillemot-right>;<A-acute>;<A-circumflex>;<A-grave>;/
<copyright>;<cent>;<yen>;<a-tilde>;<A-tilde>;<currency>;/
<eth>;<Eth>;<E-circumflex>;<E-diaresis>;<E-grave>;/
<I-acute>;<I-circumflex>;<I-diaresis>;<broken-bar>;/
<I-grave>;<O-acute>;<s-sharp>;<O-circumflex>;/
<O-grave>;<o-tilde>;<O-tilde>;<mu>;<thorn>;<Thorn>;<U-acute>;/
<U-circumflex>;<U-grave>;<y-acute>;<Y-acute>;<macron>;/
<acute>;<hyphen>;<plus-minus>;<three-quarters>;<paragraph>;/
<section>;<divide>;<cedilla>;<degree>;<diaresis>;<dot>;/
<one-superior>;<three-superior>;<two-superior>

print    <space>;<exclamation-mark>;<quotation-mark>;<number-sign>; /
<dollar-sign>;<percent-sign>;<ampersand>;<apostrophe>;     /
<left-parenthesis>;<right-parenthesis>;<asterisk>;<plus-sign>;/
<comma>;<hyphen-minus>;<period>;<slash>;/
<zero>;<one>;<two>;<three>;<four>;<five>;<six>;<seven>;/
<eight>;<nine>;<colon>;<semicolon>;<less-than-sign>;   /
<equals-sign>;<greater-than-sign>;<question-mark>;/
<commercial-at>;<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>;<Z>;/
<left-square-bracket>;<backslash>;/
<right-square-bracket>;<circumflex>;/
<underscore>;<grave-accent>;/
<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>;<z>;/
<left-brace>;<vertical-line>;<right-brace>;/
<tilde>;<C-cedilla>;<u-diaresis>;<e-acute>;<a-circumflex>;/
<a-diaresis>;<a-grave>;<a-ring>;<c-cedilla>;<e-circumflex>;/
<e-diaresis>;<e-grave>;<i-diaresis>;<i-circumflex>;/
<i-grave>;<A-diaresis>;<A-ring>;<E-acute>;<ae>;<AE>;/
<o-circumflex>;<o-diaresis>;<o-grave>;<u-circumflex>;/
<u-grave>;<y-diaresis>;<O-diaresis>;<U-diaresis>;<o-slash>;/
<sterling>;<O-slash>;<multiply>;<a-acute>;<i-acute>;/
<o-acute>;<u-acute>;<n-tilde>;<N-tilde>;<feminine>;/
<masculine>;<question-down>;<registered>;<not>;<one-half>;/
<one-quarter>;<exclamation-down>;<guillemot-left>;/
<guillemot-right>;<A-acute>;<A-circumflex>;<A-grave>;/
<copyright>;<cent>;<yen>;<a-tilde>;<A-tilde>;<currency>;/
<eth>;<Eth>;<E-circumflex>;<E-diaresis>;<E-grave>;/
<I-acute>;<I-circumflex>;<I-diaresis>;<broken-bar>;/
<I-grave>;<O-acute>;<s-sharp>;<O-circumflex>;/
<O-grave>;<o-tilde>;<O-tilde>;<mu>;<thorn>;<Thorn>;<U-acute>;/
<U-circumflex>;<U-grave>;<y-acute>;<Y-acute>;<macron>;/
```

```
<acute>;<hyphen>;<plus-minus>;<three-quarters>;<paragraph>;/
<section>;<divide>;<cedilla>;<degree>;<diaresis>;<dot>;/
<one-superior>;<three-superior>;<two-superior>

punct    <exclamation-mark>;<quotation-mark>;<number-sign>; /
<dollar-sign>;<percent-sign>;<ampersand>;<apostrophe>;     /
<left-parenthesis>;<right-parenthesis>;<asterisk>;<plus-sign>;/
<comma>;<hyphen-minus>;<period>;<slash>;/
<colon>;<semicolon>;<less-than-sign>;   /
<equals-sign>;<greater-than-sign>;<question-mark>;/
<commercial-at>;/
<left-square-bracket>;<backslash>;/
<right-square-bracket>;<circumflex>;/
<underscore>;<grave-accent>;/
<left-brace>;<vertical-line>;<right-brace>;/
<tilde>

digit    <zero>;<one>;<two>;<three>;<four>;/
<five>;<six>;<seven>;<eight>;<nine>

xdigit   <zero>;<one>;<two>;<three>;<four>;/
<five>;<six>;<seven>;<eight>;<nine>;/
<A>;<B>;<C>;<D>;<E>;<F>;/
<a>;<b>;<c>;<d>;<e>;<f>

blank    <space>;/
<tab>

toupper (<a>,<A>);(<b>,<B>);(<c>,<C>);(<d>,<D>);(<e>,<E>);/
(<f>,<F>);(<g>,<G>);(<h>,<H>);(<i>,<I>);(<j>,<J>);/
(<k>,<K>);(<l>,<L>);(<m>,<M>);(<n>,<N>);(<o>,<O>);/
(<p>,<P>);(<q>,<Q>);(<r>,<R>);(<s>,<S>);(<t>,<T>);/
(<u>,<U>);(<v>,<V>);(<w>,<W>);(<x>,<X>);(<y>,<Y>);/
(<z>,<Z>);(<a-acute>,<A-acute>);(<a-grave>,<A-grave>);/
(<a-circumflex>,<A-circumflex>);(<a-diaresis>,<A-diaresis>);/
(<a-tilde>,<A-tilde>);(<a-ring>,<A-ring>);(<ae>,<AE>);/
(<c-cedilla>,<C-cedilla>);(<eth>,<Eth>);(<e-acute>,<E-acute>);/
(<e-grave>,<E-grave>);(<e-circumflex>,<E-circumflex>);/
(<e-diaresis>,<E-diaresis>);(<i-acute>,<I-acute>);/
(<i-grave>,<I-grave>);(<i-circumflex>,<I-circumflex>);/
(<i-diaresis>,<I-diaresis>);(<n-tilde>,<N-tilde>);/
(<o-acute>,<O-acute>);(<o-grave>,<O-grave>);/
(<o-circumflex>,<O-circumflex>);(<o-diaresis>,<O-diaresis>);/
(<o-tilde>,<O-tilde>);(<o-slash>,<O-slash>);(<thorn>,<Thorn>);/
(<u-acute>,<U-acute>);(<u-grave>,<U-grave>);/
(<u-circumflex>,<U-circumflex>);(<u-diaresis>,<U-diaresis>);/
(<y-acute>,<Y-acute>);(<y-diaresis>,<Y>)

tolower (<A>,<a>);(<B>,<b>);(<C>,<c>);(<D>,<d>);(<E>,<e>);/
(<F>,<f>);(<G>,<g>);(<H>,<h>);(<I>,<i>);(<J>,<j>);/
(<K>,<k>);(<L>,<l>);(<M>,<m>);(<N>,<n>);(<O>,<o>);/
(<P>,<p>);(<Q>,<q>);(<R>,<r>);(<S>,<s>);(<T>,<t>);/
(<U>,<u>);(<V>,<v>);(<W>,<w>);(<X>,<x>);(<Y>,<y>);/
(<Z>,<z>);(<A-acute>,<a-acute>);(<A-grave>,<a-grave>);/
(<A-circumflex>,<a-circumflex>);(<A-diaresis>,<a-diaresis>);/
(<A-tilde>,<a-tilde>);(<A-ring>,<a-ring>);(<AE>,<ae>);/
(<C-cedilla>,<c-cedilla>);(<Eth>,<eth>);(<E-acute>,<e-acute>);/
(<E-grave>,<e-grave>);(<E-circumflex>,<e-circumflex>);/
(<E-diaresis>,<e-diaresis>);(<I-acute>,<i-acute>);/
(<I-grave>,<i-grave>);(<I-circumflex>,<i-circumflex>);/
(<I-diaresis>,<i-diaresis>);(<N-tilde>,<n-tilde>);/
(<O-acute>,<o-acute>);(<O-grave>,<o-grave>);/
(<O-circumflex>,<o-circumflex>);(<O-diaresis>,<o-diaresis>);/
(<O-tilde>,<o-tilde>);(<O-slash>,<o-slash>);(<Thorn>,<thorn>);/
(<U-acute>,<u-acute>);(<U-grave>,<u-grave>);/
(<U-circumflex>,<u-circumflex>);(<U-diaresis>,<u-diaresis>);/
(<Y-acute>,<y-acute>)
```

```
END LC_CTYPE


LC_COLLATE

order_start

<NUL>
<SOH>
<STX>
<ETX>
<SEL>
<tab>
<RNL>
<DEL>
<GE>
<SPS>
<RPT>
<vertical-tab>
<form-feed>
<carriage-return>
<SO>
<SI>
<DLE>
<DC1>
<DC2>
<DC3>
<RES>
<NL>
<backspace>
<POC>
<CAN>
<EM>
<UBS>
<CU1>
<IS4>
<IS3>
<IS2>
<IS1>
<DS>
<SOS>
<FS>
<WUS>
<BYP>
<newline>
<ETB>
<ESC>
<SA>
<SFE>
<SM>
<CSP>
<MFA>
<ENQ>
<ACK>
<alert>
<SYN>
<IR>
<PP>
<TRN>
<NBS>
<EOT>
<SBS>
<IT>
<RFF>
<CU3>
<DC4>
```

```
<NAK>
<SUB>
<EO>
<space>
<underscore>
<macron>
<hyphen>
<hyphen-minus>
<comma>
<semicolon>
<colon>
<exclamation-mark>
<exclamation-down>
<question-mark>
<question-down>
<slash>
<period>
<acute>
<grave-accent>
<circumflex>
<diaresis>
<tilde>
<dot>
<cedilla>
<apostrophe>
<quotation-mark>
<guillemot-left>
<guillemot-right>
<left-parenthesis>
<right-parenthesis>
<left-square-bracket>
<right-square-bracket>
<left-brace>
<right-brace>
<section>
<paragraph>
<copyright>
<registered>
<commercial-at>
<currency>
<cent>
<dollar-sign>
<sterling>
<yen>
<asterisk>
<backslash>
<ampersand>
<number-sign>
<percent-sign>
<plus-sign>
<plus-minus>
<divide>
<multiply>
<less-than-sign>
<equals-sign>
<greater-than-sign>
<not>
<vertical-line>
<broken-bar>
<degree>
<mu>
<nobreakspace>
<zero>
<one-quarter>
<one-half>
<three-quarters>
<one>
```

```
<one-superior>
<two>
<two-superior>
<three>
<three-superior>
<four>
<five>
<six>
<seven>
<eight>
<nine>
<a>
<A>
<a-acute>
<A-acute>
<feminine>
<a-grave>
<A-grave>
<a-circumflex>
<A-circumflex>
<a-ring>
<A-ring>
<a-diaresis>
<A-diaresis>
<a-tilde>
<A-tilde>
<ae>
<AE>
<b>
<B>
<c>
<C>
<c-cedilla>
<C-cedilla>
<d>
<D>
<eth>
<Eth>
<e>
<E>
<e-acute>
<E-acute>
<e-grave>
<E-grave>
<e-circumflex>
<E-circumflex>
<e-diaresis>
<E-diaresis>
<f>
<F>
<g>
<G>
<h>
<H>
<i-dotless>
<i>
<I>
<i-acute>
<I-acute>
<i-grave>
<I-grave>
<i-circumflex>
<I-circumflex>
<i-diaresis>
<I-diaresis>
<j>
<J>
```

```
<k>
<K>
<l>
<L>
<m>
<M>
<n>
<N>
<n-tilde>
<N-tilde>
<o>
<O>
<masculine>
<o-acute>
<O-acute>
<o-grave>
<O-grave>
<o-circumflex>
<O-circumflex>
<o-diaresis>
<O-diaresis>
<o-tilde>
<O-tilde>
<o-slash>
<O-slash>
<p>
<q>
<Q>
<r>
<R>
<s>
<S>
<s-sharp>
<t>
<T>
<thorn>
<Thorn>
<u>
<U>
<u-acute>
<U-acute>
<u-grave>
<U-grave>
<u-circumflex>
<U-circumflex>
<u-diaresis>
<U-diaresis>
<v>
<V>
<w>
<W>
<x>
<X>
<y>
<Y>
<y-acute>
<Y-acute>
<y-diaresis>
<z>
<Z>
UNDEFINED

order_end

END LC_COLLATE
```

```
LC_MONETARY

int_curr_symbol   "<U><S><D><space>"
currency_symbol   "<dollar-sign>"
mon_decimal_point "<period>"
mon_thousands_sep "<comma>"
mon_grouping      3
positive_sign     ""
negative_sign     "<hyphen-minus>"
int_frac_digits   2
frac_digits       2
p_cs_precedes     1
p_sep_by_space    0
n_cs_precedes     1
n_sep_by_space    0
p_sign_posn       2
n_sign_posn       2

END LC_MONETARY


LC_NUMERIC

decimal_point     "<period>"
thousands_sep     "<comma>"
grouping          3

END LC_NUMERIC


LC_TIME

abday   "<S><u><n>";/
"<M><o><n>";/
"<T><u><e>";/
"<W><e><d>";/
"<T><h><u>";/
"<F><r><i>";/
"<S><a><t>"

day     "<S><u><n><d><a><y>";/
"<M><o><n><d><a><y>";/
"<T><u><e><s><d><a><y>";/
"<W><e><d><n><e><s><d><a><y>";/
"<T><h><u><r><s><d><a><y>";/
"<F><r><i><d><a><y>";/
"<S><a><t><u><r><d><a><y>"

abmon   "<J><a><n>";/
"<F><e><b>";/
"<M><a><r>";/
"<A><p><r>";/
"<M><a><y>";/
"<J><u><n>";/
"<J><u><l>";/
"<A><u><g>";/
"<S><e><p>";/
"<O><c><t>";/
"<N><o><v>";/
"<D><e><c>"

mon     "<J><a><n><u><a><r><y>";/
"<F><e><b><r><u><a><r><y>";/
"<M><a><r><c><h>";/
"<A><p><r><i><l>";/
"<M><a><y>";/
"<J><u><n><e>";/
```

```
"<J><u><l><y>";/
"<A><u><g><u><s><t>";/
"<S><e><p><t><e><m><b><e><r>";/
"<O><c><t><o><b><e><r>";/
"<N><o><v><e><m><b><e><r>";/
"<D><e><c><e><m><b><e><r>"

d_t_fmt "%a %b %e %H:%M:%S %Z %Y"

d_fmt   "%m//%d//%y"

t_fmt   "%H:%M:%S"

am_pm   "<A><M>";"<P><M>"

END LC_TIME

LC_MESSAGES

yesexpr "[yY][eE][sS]|[yY]"
noexpr  "[nN][oO]|[nN]"
yesstr  "yes:y:Y"
nostr   "no:n:N"

END LC_MESSAGES

LC_TOD

tzdiff   0
tname    ""
dstname  ""
dststart 0,0,0,0
dstend   0,0,0,0
dstshift 0

END LC_TOD
```

Printed in U.S.A.