

# XHCS V2.0

8-Bit Code and Unicode Processing in BS2000/OSD

## **Comments... Suggestions... Corrections...**

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Feel free to send us your comments by e-mail to:

[manuals@fujitsu-siemens.com](mailto:manuals@fujitsu-siemens.com)

## **Certified documentation according to DIN EN ISO 9001:2000**

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2000.

cognitas. Gesellschaft für Technik-Dokumentation mbH

[www.cognitas.de](http://www.cognitas.de)

## **Copyright and Trademarks**

Copyright © Fujitsu Siemens Computers GmbH 2007.

All rights reserved.

Delivery subject to availability; right of technical modifications reserved.

All hardware and software names used are trademarks of their respective manufacturers.

---

# Contents

<b>1</b>	<b>Preface . . . . .</b>	<b>9</b>
<b>1.1</b>	<b>Target group . . . . .</b>	<b>9</b>
<b>1.2</b>	<b>Guide to the manual . . . . .</b>	<b>10</b>
<b>1.3</b>	<b>Changes since the last version of the manual V1.3 . . . . .</b>	<b>11</b>
<b>1.4</b>	<b>Notational conventions . . . . .</b>	<b>12</b>
<b>1.5</b>	<b>README file . . . . .</b>	<b>13</b>
<b>2</b>	<b>Introduction to XHCS . . . . .</b>	<b>15</b>
<b>2.1</b>	<b>Extended character sets . . . . .</b>	<b>16</b>
<b>2.2</b>	<b>Internationalization . . . . .</b>	<b>17</b>
<b>2.3</b>	<b>Unicode support . . . . .</b>	<b>18</b>
<b>3</b>	<b>XHCS . . . . .</b>	<b>21</b>
<b>3.1</b>	<b>Advantages of XHCS . . . . .</b>	<b>24</b>
<b>3.2</b>	<b>Requirements for XHCS . . . . .</b>	<b>25</b>
3.2.1	Hardware environment . . . . .	25
3.2.2	Software environment . . . . .	26
3.2.3	Notes on installation . . . . .	27
<b>3.3</b>	<b>Components and program interfaces . . . . .</b>	<b>28</b>
3.3.1	XHCS support via VTSU . . . . .	30
3.3.1.1	VTSU control block . . . . .	32
3.3.1.2	Terminal status . . . . .	33
3.3.1.3	Programming notes . . . . .	34
3.3.1.4	Support of special terminals via VTSU special routines . . . . .	35
3.3.2	XHCS program interfaces . . . . .	38

<b>3.4</b>	<b>Code conversion</b>	<b>39</b>
<b>3.5</b>	<b>Using extended codes</b>	<b>44</b>
3.5.1	Extended terminal protocols	44
3.5.2	Identification of file and library element codes	45
3.5.3	Activating 8-bit mode/Unicode mode	45
3.5.3.1	Explicit activation via the program	45
3.5.3.2	Implicit activation via user commands	46
3.5.3.3	Automatic activation via the system configuration	46
3.5.4	User default character set	46
3.5.5	Centralizing code tables	46
3.5.6	Applications with XHCS support	47
<b>3.6</b>	<b>XHCS with FHS and IFG</b>	<b>48</b>
3.6.1	TIAM applications	48
3.6.2	DCAM applications	50
3.6.3	openUTM applications	51
<b>3.7</b>	<b>XHCS without FHS and IFG</b>	<b>52</b>
3.7.1	TIAM applications	52
3.7.2	DCAM applications	53
3.7.3	openUTM applications	53
3.7.4	Recommendations	54
3.7.4.1	Determining the name of the data character set (CCSN)	54
3.7.4.2	Availability of XHCS	54
3.7.4.3	Detecting ISO codes	54
3.7.4.4	Terminal capabilities	54
3.7.4.5	Complete and restricted codes	55
3.7.4.6	Working in 8-bit mode	55
3.7.4.7	Working in Unicode mode	56
3.7.4.8	Working in 7-bit mode	56
<b>3.8</b>	<b>System software</b>	<b>57</b>
3.8.1	EDT V17.0	57
3.8.2	SHOW-FILE	57
3.8.3	IFG V8.3	57
3.8.4	FHS V8.3	58
3.8.5	PERCON	58
3.8.6	SORT	59
3.8.7	RSO V3.5	59
3.8.8	LMS V3.3B	59
3.8.9	FT-BS2000 V5.0	60
3.8.10	OMNIS V6.3	60
3.8.11	Examples	61

---

<b>4</b>	<b>XHCS macros</b>	<b>63</b>
<b>4.1</b>	<b>Code information: NLSCOD</b>	<b>64</b>
4.1.1	Macro call format	64
4.1.2	Operand description	64
4.1.3	Functional overview	74
4.1.4	<b>Return codes in the standard header</b>	<b>76</b>
<b>4.2</b>	<b>Compatible codes: NLSCMP</b>	<b>79</b>
4.2.1	Macro call format	79
4.2.2	<b>Operand description</b>	<b>79</b>
4.2.3	Notes on programming	83
4.2.4	<b>Return codes</b>	<b>84</b>
<b>4.3</b>	<b>String conversion: NLSCNV</b>	<b>87</b>
4.3.1	Macro call format	87
4.3.2	Operand description	89
4.3.3	Functional overview	101
4.3.4	Notes on programming	105
4.3.5	Return codes	109
<b>5</b>	<b>Definition of code tables</b>	<b>113</b>
<b>5.1</b>	<b>Code names</b>	<b>113</b>
5.1.1	System default code	113
5.1.2	User default character set	114
<b>5.2</b>	<b>Grouping compatible codes</b>	<b>115</b>
<b>5.3</b>	<b>Table structure</b>	<b>116</b>
5.3.1	Table for converting to the reference code	116
5.3.2	Table for converting lowercase letters to uppercase	117
5.3.3	Sort table	117
5.3.4	Table of character properties	117
5.3.5	Table for converting reference codes	118
5.3.6	Table for Unicode mapping	118
<b>5.4</b>	<b>Creation and modification of code tables</b>	<b>119</b>
5.4.1	Generating sets of tables	120
5.4.2	NLSHEAD and NLSCCS macros	121
5.4.3	Example of the generation of a set of tables	123
5.4.4	Installation of user-defined tables	128
<b>5.5</b>	<b>Summary of rules and conventions</b>	<b>129</b>

<b>5.6</b>	<b>Defining additional Unicode characters</b>	<b>130</b>
5.6.1	Defining pseudo 8-bit code	130
5.6.2	Defining sort information	131
5.6.3	Assembling GNLMTAB and linking in an extended code base	131
<b>5.7</b>	<b>Adding new coded character sets dynamically: ADD-CODE-TABLES</b>	<b>132</b>
<b>5.8</b>	<b>Arabic and Persian codes</b>	<b>134</b>
5.8.1	Supported codes	134
5.8.2	Rules for Arabic codes	135
<b>5.9</b>	<b>Support of the euro symbol in XHCS</b>	<b>136</b>
5.9.1	Use of ISO code variants ISO 8859-1/-2/-7/-9	136
5.9.2	Introduction of the ISO code variant ISO 8859-15	137
5.9.2.1	Compatibility restrictions with regard to Arabic code variant F	138
<b>6</b>	<b>Preparations for use</b>	<b>139</b>
<b>6.1</b>	<b>Preparations for XHCS support</b>	<b>139</b>
6.1.1	PDN generation	139
6.1.2	XHCS	140
6.1.3	Activating the 8-bit environment for 8-bit data display terminals	140
<b>6.2</b>	<b>Preparation for VTSU special routines</b>	<b>141</b>
6.2.1	PDN generation	141
6.2.2	VTSU	141
6.2.3	Activating the 8-bit environment for Arabic/Persian terminals	141
6.2.4	Activating European 7-bit data display terminals	141
6.2.5	Activating ESC printers	142
<b>7</b>	<b>Tables</b>	<b>143</b>
<b>7.1</b>	<b>Standard CCSN tables</b>	<b>143</b>
<b>7.2</b>	<b>Overview of XHCS VTSUCB return information</b>	<b>146</b>
<b>7.3</b>	<b>Supported line codes and BS2000 EBCDIC codes</b>	<b>147</b>
7.3.1	Supported line codes and BS2000 EBCDIC codes for European alphabets	147
7.3.1.1	8-bit line codes and associated EBCDIC codes for European alphabets	148
7.3.1.2	7-bit line codes and associated EBCDIC codes for European alphabets	170
7.3.2	Supported line codes and BS2000 EBCDIC codes for Arabic alphabets	184
7.3.3	Supported line codes and BS2000 EBCDIC codes for Persian alphabets	191

**Glossary . . . . . 195**

**Abbreviations . . . . . 199**

**Related publications . . . . . 201**

**Index . . . . . 205**





---

# 1 Preface

XHCS (**Extended Host Code Support**) is a character-handling software product for the BS2000/OSD operating system. It allows BS2000/OSD to display all national languages defined in the international code tables according to ISO 8859. These languages comprise all Western and Eastern European languages including Cyrillic, Greek and the Baltic languages, as well as Arabic, Persian (Farsi) and Maltese. XHCS also supports the Unicode character set.

## 1.1 Target group

This manual is aimed at users of the DCAM, TIAM and openUTM access methods who wish to enjoy the advantages of having an expanded standard character set and/or the Unicode character set in their programs. It is also intended for systems support staff who wish to tailor the predefined code tables to their requirements.

Previous knowledge is required of BS2000, the DCAM, TIAM, openUTM access methods, the VTSU software component and, where applicable, FHS and IFG. Systems support staff must also be familiar with systems support for BS2000/OSD-BC.

Under [Related publications](#) at the back of this manual you will find suggestions for further reading. In addition to the code tables described in this manual, those of the supported terminals and printers are also required.

## 1.2 Guide to the manual

This manual describes how to use extended code tables, support of the Unicode character set, and how to create and modify code tables.

The [chapter “Introduction to XHCS”](#)

provides a brief overview, explains extended character sets, describes the special features of Unicode support in XHCS, and shows how XHCS provides for international application use.

The [chapter “XHCS”](#)

describes the concept and advantages of XHCS, the components and program interfaces, and explains the practical use of XHCS.

The [chapter “XHCS macros”](#)

contains a detailed description of the XHCS macro.

The [chapter “Definition of code tables”](#)

is aimed at systems support staff. It explains how to create and modify code tables.

The [chapter “Preparations for use”](#)

gives information regarding preparations for use.

The [chapter “Tables”](#)

contains all important tables, such as supported line codes and BS2000 EBCDIC codes.

At the back of the manual you will find a glossary, a list of abbreviations, related publications, and an index.

## 1.3 Changes since the last version of the manual V1.3

This manual contains the following changes and functional extensions for XHCS V2.0 compared to V1.3:

Section		As of version
2.3	XHCS supports the Unicode character set:	V2.0
3.3	<ul style="list-style-type: none"> <li>● Extension of the GNLMTAB module because of Unicode.</li> </ul>	
3.3.1	<ul style="list-style-type: none"> <li>● Support of an 8-bit terminal or of the terminal emulation MT9750 in Unicode mode</li> </ul>	
4.1	<ul style="list-style-type: none"> <li>● The NLSCOD program interface can output conversion tables for conversion between 8-bit codes and Unicode.</li> </ul>	
4.1	<ul style="list-style-type: none"> <li>● The NLSCOD program interface for outputting uppercase/lowercase, properties and sort tables has been extended by Unicode tables.</li> </ul>	
4.1	<ul style="list-style-type: none"> <li>● The NLSCOD program interface can output a table with the byte properties for UTFE.</li> </ul>	
4.2	<ul style="list-style-type: none"> <li>● The NLSCMP program interface supplies information on the output code.</li> </ul>	
4.2 / 4.3	<ul style="list-style-type: none"> <li>● At the NLSCMP and NLSCNV interfaces you can also specify and receive the input and output strings using an address and a length.</li> </ul>	
4.3	<ul style="list-style-type: none"> <li>● The NLSCNV program interface offers the option of specifying a Unicode variant as the source or target code for converting strings.</li> </ul>	
4.3	<ul style="list-style-type: none"> <li>● The NLSCNV program interface can be called from TU via a call interface without SVC.</li> </ul>	
4.3	<ul style="list-style-type: none"> <li>● All 7-/8-bit character sets can be converted in an unambiguously reversible manner.</li> </ul>	
4.3	<ul style="list-style-type: none"> <li>● With NLSCNV, C'.' (U+002E) is now used as the wildcard character, not the NUL character.</li> </ul>	
5.6	<ul style="list-style-type: none"> <li>● XHCS supports the definition of additional Unicode characters.</li> </ul>	
5.7	The BS2000 command /ADD-CODE-TABLES enables you to modify and extend code tables dynamically during ongoing operation.	V1.4
7.3.1.1	XHCS supports the ISO code variant 8859-4.	V2.0
7.3.1.1	XHCS supports the ISO code variant 8859-3.	V2.0

## 1.4 Notational conventions

To make this manual easier to read, it uses a number of notational conventions also used in many other BS2000 manuals. They are explained in the table below.

Presentation of syntax	Syntax definition	Example
UPPERCASE LETTERS	Used for constants which the user must enter in this form.	"YES"
lowercase letters	Used for variables the contents of which can vary. The user must enter the current values.	partner-name
/	Alternate entries are separated by a slash.	MF=L / <u>S</u>
[ ]	Brackets enclose entries that can be omitted. When there is a comma inside the brackets, it need only be entered if the optional entry is specified. A comma outside the brackets must be specified, even when the optional entry is not used.	[password4] filename[,ERASE]
<u>underscored</u>	Default values are underscored. The default is the value the system uses when the user makes no entry.	INFO= <u>*LIST-OF-CCS</u>
...	An ellipsis indicates that the previous item can be repeated any number of times.	(archive-no,...)
()	An expression used to describe a variable appears in parentheses. This indicates the range of values at a glance. Several characters are required to describe a variable, and the parentheses form them into a unit.	(0 < length < 9)
<b>i</b>	Indicates particularly important information.	

## 1.5 README file

Information on functional changes and additions to the current product version described in this manual can be found in the product-specific README file. You will find the README file on your BS2000 computer under the file name SYSRME.XHCS-SYS.020.E.

The user ID under which the README file is cataloged can be obtained from your system support. You can view the README file using the /SHOW-FILE command or an editor, and print it out on a standard printer using the following command:

```
/PRINT-DOCUMENT filename, LINE-SPACING=*BY-EBCDIC-CONTROL
```

SPOOL version smaller than 3.0A:

```
/PRINT-FILE FILE-NAME=filename, LAYOUT-CONTROL=  
PARAMETERS(CONTROL-CHARACTERS=EBCDIC)
```



---

## 2 Introduction to XHCS

The XHCS (Extended Host Code Support) software product is the central information source for coded character sets (CCS) available in BS2000/OSD. Programs do not have to permanently store information about character sets; instead, they receive this information from XHCS.

XHCS identifies data codes, regardless of their source, whether they come from a terminal input, program output, or from another system. The character set name (CCSN, Coded Character Set Name) identifies the transferred data code. XHCS provides coded character sets in the form of tables. Depending on user-specific requirements, existing character sets can be adapted to local requirements and the user's own character sets can be added to existing character sets.

With XHCS, characters from extended character sets can be quickly, easily and reliably transferred between the computer and peripherals. Since the available 95 characters (in 7-bit code) are expanded to 189 characters (in 8-bit code) in each character set, these are known as "extended character sets."

For the 8-bit ISO8859 codes which are supported, XHCS also supports the Unicode character set and also approx. 70 characters from the Unicode character set which are used in reporting in accordance with the international guideline. In addition, XHCS also permits the character set to be extended for Unicode support.

## 2.1 Extended character sets

Extended character sets are also referred to as “8-bit codes” or extended codes because the number of available code positions has been doubled. The number of code positions allows you to define characters for different languages that are still compatible with the code originally used, the ASCII code (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange). The different 8-bit codes are defined in the international ISO 8859 standard. In the “left” (lower value) half of the code table, they all have a common part, similar to ASCII, while in the “right” (higher value) half they are different. Individual codes are combined into groups of compatible codes that are identified via their ISO code variant number. The following codes are currently defined as standard in ISO 8859:

- 8859-1 Latin-1 (Western and Northern Europe)
- 8859-2 Latin-2 (Eastern Europe, excluding Turkey and the Baltic States)
- 8859-3 Latin-3 (Mediterranean region and South Africa)
- 8859-4 Latin-4 (Scandinavia and the Baltic States)
- 8859-5 Cyrillic
- 8859-6 Arabic
- 8859-7 Greek
- 8859-8 Hebrew
- 8859-9 Latin-5 (Turkey, Western Europe including Scandinavia)
- 8859-10 Latin-6 (Northern Europe and the Baltic States)
- 8859-15 Latin-9 (Western and Northern Europe including the euro symbol €)

[Chapter “Tables” on page 143](#) provides an overview of all codes supported by XHCS and their assignment to EBCDIC codes.

On the BS2000 side, the ISO 8859-1 character set is represented by EBCDIC.DF.04-1. The EBCDIC (**E**xtended **B**inary **C**oded **D**ecimal **I**nterchange **C**ode) that BS2000 uses must be extended so that every character has a counterpart in the corresponding ISO 8859-x. Since EBCDIC codes are not standardized, different assignments exist between EBCDIC and ISO codes. For instance, the Cyrillic alphabet defined in part 5 of the ISO 8859 standard (ISO 8859-5) has two different equivalent computer codes: EBCDIC.DF.04-5 and EBCDIC.EHC.LC.

The defined assignment between ISO 8859-1 and EBCDIC.DF.04-1 plays a special role. An EBCDIC table is defined for every ISO table. Each EBCDIC table contains the same characters but with a different code assignment. These EBCDIC tables are identified as EDF041 through EDF049 and EDF04F. They were selected so that the same conversion table can be used for conversions from ISO 8859-n ( $n=1..9$  or  $n=15$ ) to EDF04n ( $n=1..9$  or  $n=F$ ) and vice versa. This standard conversion is used by network components and file transfer programs. In addition to the use of existing extended character sets, XHCS allows systems support staff to define the coding of extended character sets, select an extended character set as the system standard, and assign a user standard code to each user.



## 2.2 Internationalization

Today, over 2,000 languages are spoken in the world. Many of these languages are common only in one specific country, while other languages are extend over several countries. In addition, more than one language is spoken in quite a few countries.

For software products to be used on an international basis, they not only have to take into consideration the language used at the location and the characters in the associated alphabet, but they also have to support several languages in multi-language countries. In order to establish business connections with other countries, they also have to provide support for the alphabets of these countries.

XHCS makes it possible to display the alphabets of all national languages in BS2000 providing they are defined in international code tables according to ISO 8859. This includes the character sets of all Western and Eastern European languages including Cyrillic, Greek and the Baltic languages, as well as Arabic, Persian (Farsi) and Maltese.

For the standards defined in ISO 8859, extended 8-bit coding can display the characters in the different character sets. Only single-byte code processing is possible for letter alphabets. Single-byte code processing cannot display ideographic or concept alphabets such as the different Asian alphabets. In these cases, multibyte code processing is required. National standards are established for this in each of the associated countries (China, Taiwan, South Korea, Japan).

## 2.3 Unicode support

XHCS V2.0 supports the following Unicode variants:

- UTF-16  
is largely identical to the two-byte representation UCS-2. Surrogate pairs (which permit two-byte representation of characters between x'FFFF' and x'10FFFF') are not supported.  
XHCS notation: UTF16 or 'UNICODE'.
- UTF-8  
is the most widespread Unicode variant.  
XHCS notation: UTF8.
- UTF-EBCDIC  
is a special implementation of the Unicode variant originally developed for IBM (see <http://www.unicode.org/reports/tr16/>). Unlike this variant, XHCS uses the EDF041 table to map to EBCDIC (see [page 149](#)).  
XHCS notation: UTFE.



When Unicode is referred to in this manual, the two-byte code UTF-16 is always meant. Unicode variant can mean both UTF-16 and UTF-8 or UTF-EBCDIC.

For detailed information on the various Unicode variants, please refer to the overview manual “Unicode in BS2000/OSD” [[32](#)].

XHCS V2.0 includes conversion tables for converting the 8-bit ISO codes supported (ISO8859-1/2/3/4/5/7/9/15) to Unicode and vice versa.

In addition to the characters which are contained in these ISO codes, XHCS also supports approx. 70 characters of the Unicode character set which are used by our customers (see also the [section “Defining additional Unicode characters” on page 130](#)). This means that only the Unicode positions which correspond to these characters are defined in the conversion tables. A pseudo 8-bit ISO code has been defined for further characters, see also the [section “Defining pseudo 8-bit code” on page 130](#).

The lowercase/uppercase table, the properties table and the sort table for Unicode are only defined for characters from the ISO codes which are supported and additional characters.

In addition, you can extend the character set for Unicode support: to do this, define the relevant code tables containing these new characters (see [page 130](#)).

## Normalization

A special feature of the Unicode variant UTF-16 is the normalization of Unicode strings (see also <http://www.unicode.org/reports/tr15/>).

XHCS supports the following slightly restricted functionality:

- Composition  
A basic character is combined with one or two following diacritical characters to form a single character (if defined).  
Macro call: NLSCNV ACTION=COMPOSE,....
- Decomposition  
A composed character is decomposed into the basic character and one or two following diacritical characters (if defined).  
Macro call: NLSCNV ACTION=DECOMPOSE,....

The basis for normalization is provided by the normalization table which is derived from the Unicode sort table and ensures high performance when obtaining information.

For detailed information on the various Unicode variants, please refer to the overview manual “Unicode in BS2000/OSD” [32].



---

## 3 XHCS

XHCS implements the new character handling concept in BS2000/OSD. It allows different character sets to be used, and it makes all character-processing component mechanisms available to detect and interpret current character sets.

The XHCS concept includes:

- Extended terminal device protocols

Terminals have all European alphabets; via an extended terminal device protocol, they inform the system of the currently defined character set and are able to dynamically change this setting upon request. If system requirements demand it, they switch back into the mode of a conventional terminal.

- Expanded BS2000 system components

Providing they implement character-related operations such as sorting or recoding, they can identify the current data coding and the terminal setting and adjust their processing accordingly. During this process, they take advantage of central help from XHCS.

- The XHCS subsystem

XHCS provides all information on all character sets that is required for comparison and conversion operations. This means that the remaining components no longer have to provide the corresponding tables themselves. The XHCS interfaces are also available to each application program.

The following figure illustrates XHCS in its software environment.

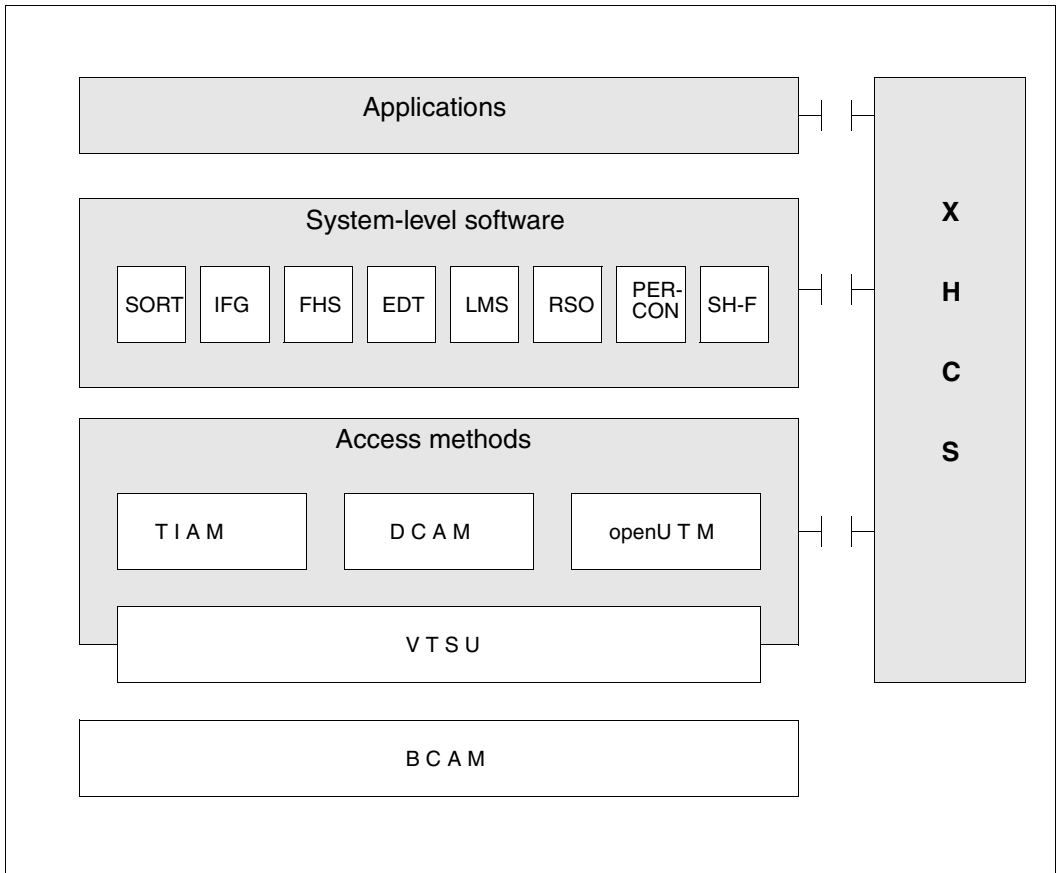


Figure 1: XHCS in the software environment

The TIAM, DCAM and openUTM access methods are linked to XHCS via the software component VTSU. Users can access XHCS using the corresponding TIAM, DCAM or openUTM application programs. However, openUTM users can only access XHCS via FHS. DCAM and TIAM users can access XHCS ‘directly’ (via the VTSUCB) or via FHS.

The XHCS product is a source of information regarding coded character sets available in the system. XHCS provides program interfaces for the following functions:

- supply and provision of different tables of a defined code (conversion into another code, conversion of lowercase to uppercase letters, table of sorting priorities and different character properties)
- direct conversion of character strings
- supply of information about the existing codes in the system and possibilities of conversion

With these interfaces, applications can be operated independently of the existing code. XHCS offers a tool for software internationalization insofar as character sets are affected.

The codes in XHCS are defined with the help of adapted tables. They can add coded character sets to the predefined character sets and adapt them to local requirements, e.g. change the sort sequence of a code's characters.

Codes are logically combined into groups of compatible codes, depending on the contained character set. Code groups are identified by their ISO variant number which refers to one of the variants of the ISO 8859 standard. The group of the Unicode variants is an exception here: a pseudo variant number has been introduced for this group, see also the [section "Code information: NLSCOD" on page 64](#). All 7-/8-bit codes count as partially compatible with Unicode.

All codes of an existing variant contain the same character set as the corresponding ISO 8859 character set or a subset.

Conversions can only occur between codes in the same group since XHCS does not recognize the equivalent characters of another group.

### 3.1 Advantages of XHCS

XHCS offers the following advantages:

- Complete and clear display of a language's characters  
By doubling the usable code positions, XHCS provides for the clear display of all national and international characters defined in ISO 8859.
- Display of foreign language data  
Foreign language texts, names and addresses can be rendered clearly and correctly.
- Data exchange with partner systems  
Comprehensive system communications products such as emulation or file transfer carry out code conversions at the system boundaries to provide for correct processing in the target system.
- Support for multilingual applications  
With XHCS, applications can easily be ported to other languages or country variants, and different languages and databases can be edited simultaneously.
- Individual migration concept  
XHCS ensures the complete integrity of the existing hardware and software configuration. You can determine whether the existing data should be mixed with new data or whether it should be maintained separately and if the existing applications should be used to process the new data or when they should be adapted.



## 3.2 Requirements for XHCS

This section describes the hardware and software requirements for the use of XHCS. It also contains information about installation and the support of special data display terminals.

### 3.2.1 Hardware environment

XHCS supports the following 8-bit capable terminals:

#### Data display terminals

- 9756-National (European, Arabic, Persian)
- 9758-M486
- 9759-M2/M4
- 9763-M/C/G
- EMDS V5.1 emulation (SINIX)
- MT9750 V7.0C emulation (MS-Windows)
- DOORS emulation Winsock V3.1A
- WT9750 V2.0A

Using an extended terminal protocol, these data display terminals can transfer the character set used to the system and dynamically modify the character set.

To work in 8-bit mode, data display terminal 9756-National has to be entered in the PDN type DSS-9755, while other data display terminals have to be entered as type DSS-9763.

For EMDS, the corresponding environment variables have to be correctly set (refer to the “EMDS (SINIX)” manual [10]).

#### Printers

- 9001-32
- 9011-28/29
- 9012
- 9013-31x
- 9014
- 9097
- 9021
- 4819/20

All printers have to be entered in the PDN using their real names.

## 3.2.2 Software environment

### XHCS

XHCS is supplied with predefined codes (object modules) that can be used directly. These codes and others are also available as source code (assembler macros), which systems support staff can use to create their own XHCS table modules.

### BS2000

XHCS supports the following functions:

- a CCSN is permanently assigned to all files and library elements
- a standard user CCSN can be assigned to each user. This provides CCS with the information necessary to use 8-bit mode for inputs and outputs
- VTSU special routines

### VTSU

System software supporting virtual terminals (VTSU) is a part of *openNet* Server (previously DCAM). The information in this description applies to VTSU as of V13.2A.

The support of extended codes is a standard function of VTSU. VTSU special routines (see [page 35](#)) must be used in the following special cases:

- 7-bit code support for national 7-bit terminals
- 8-bit alphabet support for Arabic, North African (Arabic/French) and Farsi (Persian)
- support of Unicode
- support of ESC printers 9001, 9013 and 9022

### TIAM

Unicode is supported in TIAM V13.1C and higher.

The information which VTSU has received from a terminal or an emulation is displayed using the BS2000 command `SHOW-TERMINAL-ATTRIBUTES INF=*ALL`. This information enables you to determine whether the terminal is Unicode-capable.

### PDN

PDN as of V11.0A is required for full support of VTSU as of V11.0.

PDN has become less important. However, for the sake of clarity, the descriptions and explanations relating to PDN have been retained in this manual.

### 3.2.3 Notes on installation

The optional product XHCS is a dynamic subsystem loaded via DSSM (Dynamic SubSystem Management); it cannot be unloaded. The XHCS-SYS subsystem therefore has to be declared in the subsystem catalog (see the BS2000/OSD-BC “System Installation“ manual [3]). By default, the module library SYSLNK.XHCS-SYS.*xxx* (*xxx* = version number) and the REP files SYSREP.XHCS-SYS.*xxx* (*xxx* = version number) must be cataloged under TSOS or installed using IMON.

The SYSSSC.XHCS-SYS.*xxx* file (*xxx* = version number) contains the subsystem declarations (SSD-OBJ file). By default, the subsystem declarations expect the files to be under TSOS or somewhere where they were loaded using IMON.

### 3.3 Components and program interfaces

XHCS is defined as a dynamic subsystem (called XHCS-SYS). It is managed by DSSM (Dynamic SubSystem Management) and loaded when the system is started. XHCS is available to both privileged and nonprivileged programs. It is based on tables, all of which describe codes. Each code described is assigned several tables. These tables are created and modified using macros. They can be accessed both by the operating system and application programs. The code tables are contained in the GNLMTAB module. If the standard codes of the GNLMTAB module are not to be used, the modified module must be assembled once before the subsystem is loaded and stored in the SYSLNK.XHCS-SYS.020 module library. XHCS is shipped with a standard GNLMTAB module, ready for immediate use, which allows you to work with the following ISO 8859 codes:

- Standard 1: EDF041, ISO88591, EDF04DRV, EDF03IRV, ISO646
- Standard 2: EDF042, EEHCL2, ISO88592
- Standard 3: EDF043, ISO88593
- Standard 4: EDF044, ISO88594
- Standard 5: EDF045, EEHCLC, ISO88595, EEHCLC1
- Standard 7: EDF047, EEHCLG, ISO88597
- Standard 9: EDF049, ISO88599
- Standard F: EDF04F, ISO8859F, WCP1252P

The group reference code (see [section “Grouping compatible codes” on page 115](#)) occupies the first position in every group. EDF03IRV is the system standard.

The GNLMTAB source code is in the SYSSRC.XHCS-SYS.020.GNLMTAB library and also contains, as comments, the code lines defining the remaining standard character sets listed in the [chapter “Tables” on page 143](#). If you wish to use one or more of these character sets, systems support staff must first delete the comment characters of the relevant code lines and the source code must then be reassembled. Systems support staff can also create the GNLMTAB module themselves using the NLSCTAB macro or the NLSHEAD and NLSCCS macro (see [page 120ff](#)).

In XHCS V2.0 and higher, the GNLMTAB module also contains the following:

- The Unicode sorting information
- The Unicode properties table
- The table with the assignments of uppercase letters to the corresponding lowercase letters and vice versa for Unicode
- The table for converting uppercase letters to the corresponding lowercase letters for the UTFE one-byte codes
- The table for converting lowercase letters to the corresponding uppercase letters for the UTFE one-byte codes
- The table with the byte properties for UTFE

### 3.3.1 XHCS support via VTSU

The software component VTSU implements a virtual line/page terminal. VTSU allows users of the DCAM, TIAM and openUTM access methods to use logical (i.e. symbolic) control characters instead of device-specific real control characters when editing messages. All logical control characters, together with their symbolic names, are contained in a data structure that you can copy to the application program as required. VTSU converts these logical control characters to the corresponding real control characters required by the relevant terminal. In this way, VTSU makes application programs independent of the type of real terminal used.

In supporting XHCS, VTSU distinguishes between the following:

- 8-bit terminals in 8-bit mode that send either a 7-bit code with SI/SO control characters or an 8-bit code
- 8-bit terminals in 7-bit mode that send a 7-bit code
- 8-bit terminals in Unicode mode that send a Unicode
- 7-bit terminals that send a 7-bit code

The message header is always coded in an ISO code.

#### **8-bit data display terminals in 8-bit mode**

To support 8-bit data display terminals, VTSU needs to know the terminal type, which ISO code variants the system supports, whether a code extension mechanism (SI/SO) is used, and whether the codes used are compatible with the codes supported by the data display terminal. VTSU obtains this information by means of a status query and a compatibility check. With TIAM and openUTM applications, the status is queried automatically when the connection is established. With DCAM applications, you have to query the status yourself (OPTCD = TERMSTAT during the YOPNCON call).

Based on the information obtained in the status query, VTSU establishes whether the computer is connected with 7- or 8-bit data display terminals. If the status is not queried, the extended standard code is ignored. In this case, EDF03IRV is the default code.

The compatibility check takes place during LOGON processing for TIAM applications and on connection setup for DCAM and openUTM applications. If the terminal does not support the specified code, the user default code is ignored; in this case, too, the 7-bit code is used. TIAM applications issue a warning if this occurs.

XHCS is not called until VTSU has established which terminal type and code are being used. XHCS provides the conversion tables required to convert to or from the specified code.



- If, when using a DCAM application, you do not query the status, the 9758 data display terminal is treated as a 9763. This leads to errors in formatted mode (FHS, LINE mode, extended line mode). If you query the status, the 9758 data display terminal is treated as a 9755, even if it was generated in PDN as a 9763. Depending on the PDN generation of the 9758 data display terminal, the following situations can occur:

Definition of the 9758 data display terminal in PDN	FHS response
The 9758 data display terminal is generated in PDN as a 9763. The status is not queried.	Formatting unsuccessful; FHS formatting is carried out for a 9763 instead of a 9755.
The 9758 data display terminal is generated in PDN as a 9763. The status is queried.	Formatting successful; FHS formatting is carried out for a 9755.
The 9758 data display terminal is generated in PDN as a 9755 or 9758.	Formatting successful; FHS formatting is carried out without any problems. However, the 9758 data display terminal cannot be used in 8-bit mode.

- The ICE character set cannot be used in 8-bit mode.

### 8-bit terminal in Unicode mode

VTSU supports Unicode as of VTSUB V13.2A. When Unicode data is to be received or sent, VTSU must be notified of this. This can be done in the applications by means of the VTSU-B control block or using the BS2000 command /MODIFY-TERMINAL-OPTIONS. The setting in the VTSU-B control block has priority over the value set with /MODIFY-TERMINAL-OPTIONS. However, it only applies for the current call.

In order to operate with the terminal emulation MT9750 V7.0 in Unicode mode, the emulation's session parameters must be configured appropriately (see also [“Terminal emulation MT9750 in Unicode mode” on page 37](#)).

### 7-bit terminal/8-bit terminal in 7-bit mode

7-bit data display terminals and 8-bit data display terminals in 7-bit mode have the same code handling. This means that when one of these data display terminals is connected, the codes are converted exclusively by the PDN network software. VTSU ignores the control characters SI and SO and the ISO code variants.

Users and systems support staff must ensure that the extended standard code matches the keyboard variant.

## 8-bit printers

7- and 8-bit printers have the same PDN generation and, even if a status query is carried out, VTSU cannot tell which printer type is being used. The printer type (7-bit/8-bit) and link type (7-bit/8-bit) are therefore set by means of operating parameters (VTSU) or the free text parameter (PDN). Each printer has its own operating parameters to describe the link and printer types. For a description of how to set the operating parameters, please refer to the “VTSU“ manual [1].

To carry out code conversion, VTSU must take the printer control escape sequences out of the message. For VTSU to recognize these sequences, they must be coded in the EBCDIC.DF.03 standard. In physical mode (MODE=PHYS), however, you can use special escape sequences that do not belong to the EBCDIC kernel. VTSU does not recognize these special escape sequences and attempts to convert them on the basis of the defined character set. This can lead to code conversion inconsistencies. The parameter CODETR=NO in the VTSU control block can be used to prevent VTSU from carrying out code conversion.

When converting codes, VTSU assumes automatically that the printer’s character set corresponds to the required 8-bit character set.

### 3.3.1.1 VTSU control block

The VTSU control block (VTSUCB) allows you to set special VTSU parameters for input and output message editing, independently of the access method used. However, this is not possible for openUTM applications. You can also use the XHCS functionality. The VTSU control block is passed to VTSU for the input and output operations of the corresponding access methods.

In the VTSU control block, you can use the CCSNAME parameter to specify the name of the character set to be used. The possible values of CCSNAME are either a CCSN or the value \*EXTEND, which identifies the extended user default character set. The code must be an extended EBCDIC. If you enter \*EXTEND, VTSU uses the extended user default character set for current processing. You can ascertain the name of the character set via the DCSTA interface. If you specify neither \*EXTEND nor the extended user default code name, the terminal remains in 7-bit mode unless you have set it to 8-bit mode (e.g. by entering MODE=PHYS or the TIAM command MODIFY-TERMINAL-OPTIONS).

You can also set Unicode mode by means of the macro call VTSUCB. However, only the Unicode variant UTFE can be specified.

The specified code is used only for the current call. This means that, in the case of a 7-bit connection, the control characters SI and SO are interpreted (input) and inserted (output) only for this call, and conversion to or from the specified code is prepared only for this call.

This type of processing is only possible with MODE=MIXED, LINE, EXTEND, INFO, FORM and PHYS, not with MODE=CHIP or MODE=TRANS.



The VTSU control block is available for the DCAM and TIAM access methods in the ASSEMBLER, FORTRAN, PL/1, C and COBOL programming languages.

The VTSUCB can be used to specify the code for the following input/output operations:

Access method	Language	Input/output operations
TIAM	Assembler	WROUT,RDATA,WRTRD
TIAM	COBOL	WROUT,RDATA,WRTRD (CALLs)
TIAM	FORTRAN	WROUT,RDATA,WRTRD (CALLs)
TIAM	PL/I	WROUT,RDATA,WRTRD (CALLs)
TIAM	C	WROUT,RDATA,WRTRD (CALLs)
DCAM	Assembler	YSEND,(YRECEIVE),YSENDREC
DCAM	COBOL	YSEND,(YRECEIVE) (CALLs)

### 3.3.1.2 Terminal status

Before 8-bit inputs/outputs or Unicode inputs/outputs can be processed, you must check which options the terminal has. The inputs/outputs can come from a terminal or a terminal emulation.

You can use the VTSU macro DCSTA (TYPE=BASIC) to obtain information about the relevant characteristics of the terminal and connection. The information is updated by calling the TSTAT macro (TIAM) or the YINQUIRE macro (DCAM).

The following DSTA macro fields relate to XHCS:

STATTYPE	Indicates whether the terminal can operate in 8-bit mode
STACCSNN	Number of supported 8-bit/Unicode character sets
STACSS1-16	Numbers of the supported 8-bit/Unicode character sets
STACURCH	Contains the name of the user default character set if the terminal is 8-bit-capable and the character set is compatible with the terminal.
STAACTCH	Contains the name of the active extended character set.

These interfaces are also available for COBOL, C, PL/1 and FORTRAN.

In addition to using the VTSU macro DCSTA, you can also use the BS2000 command SHOW-TERMINAL-ATTRIBUTES to display information on all the attributes of your terminal. This command enables you to determine the mode (e.g. 7-bit, 8-bit, Unicode) in which your terminal or emulation is operating. Detailed information on this subject is provided in the descriptions of the BS2000 commands [5].

### 3.3.1.3 Programming notes

- If XHCS is not called by means of the VTSU control block or the MODIFY-TERMINAL-OPTIONS command, VTSU returns automatically to 7-bit mode. The subsequent message processing is therefore in 7-bit mode again. When the mode changes in this way, the screen is deleted.
- For code conversion, VTSU must remove the printer control escape sequences from the message. These sequences must be coded in the EBCDIC.DF.03 standard for VTSU to recognize them. However, when you work in physical mode (MODE=PHYS) with printers, you can use special escape sequences that do not belong to the EBCDIC kernel. VTSU does not recognize these special escape sequences and attempts to convert them on the basis of the specified character set. This can lead to code conversion inconsistencies. To suppress VTSU code conversion in this case, the parameter CODETR=NO is offered in the VTSU control block.
- The name of the character set to be used is specified in the VTSU control block. You can enter blanks (the default, specifying that the VTSU control block will not be used), \*EXTEND (which specifies that the extended system default code be used) or the extended standard code name.
- A blank code name has exactly the same effect as an unused VTSU control block (i.e. the connected terminal remains in 7-bit mode and VTSU does not carry out code conversion). If, however, the terminal is set to 8-bit mode using the MODIFY-TERMINAL-OPTIONS command, conversion always takes place.
- If there is a change from one character set to another, the screen is automatically deleted. If MODE=LINE or MODE=EXTEND applies, you are prompted to confirm a mode switch. This allows you to read the screen contents before they are deleted. It is possible to switch from a 7-bit mode to an 8-bit mode, from an 8-bit mode to a 7-bit mode or from one 8-bit mode to another.
- If XHCS is not loaded, all messages are processed as 7-bit messages. If the XHCS functions CODETR and CCSNAME are called even although XHCS is not loaded, the call is rejected in the VTSU control block with an appropriate error message.
- When specifying the character set name, make sure that you always specify the EBCDIC variant of the code name. The name of the corresponding ISO code variant is rejected automatically. You must also specify the code names of fully compatible codes, since codes with limited compatibility are rejected.

### 3.3.1.4 Support of special terminals via VTSU special routines

European 7-bit data display terminals, Arabic/Persian 8-bit data display terminals, and Escape printers are special terminals that have to be supported by means of VTSU special routines.

Special routines are called internally by VTSU; they are supplied with VTSU and can be integrated into the entire system.

For a detailed description of VTSU special routines, the configuration file, and the installation procedure, refer to the “VTSU“ manual [1].

#### System environment for the VTSU special routines

VTSU-B as of V11.0A

#### European 7-bit terminals

By default, XHCS supports only 8-bit stations. National 7-bit devices are supported using the “national 7-bit support” function in VTSU-B.

The following 7-bit code variants are possible:

- EBCDIC.NHC.SWE (Swedish 7-bit code)
- EBCDIC.NHC.HUN (Hungarian 7-bit code)
- EBCDIC.NHC.CYR (Cyrillic 7-bit code)
- EBCDIC.NHC.GRE (Greek 7-bit code)

Devices and the desired code variants are defined in a configuration file that must be created by the user. The configuration file must be a SAM file and have the name “SYSPAR.VTSU-B.*xxx*.CONFIG“ (*xxx* = version number). It must be cataloged under TSOS or installed with IMON. European 7-bit data display terminal support is activated via an installation procedure. The installation procedure is supplied with VTSU-B.

#### Escape printers

As with European 7-bit data display terminals, Escape printers have to be defined in the configuration file using their station name, computer name and variant. The installation procedure does not have to be called to support Escape printers. Escape printers do have to be defined as 8-bit printers using the PDN free text parameter (PDN as of V11) or in the corresponding VTSU operating parameter file.



#### CAUTION!

The PDN free text parameter has a higher priority than the corresponding VTSU operating parameters.

### Arabic/Persian 8-bit data display terminals

The “standard XHCS” does not support Arabic or Persian codes. The “ARA-/FAR-/NAF support” function in VTSU-B can be used to make Arabic/Persian/Arabic-French codes available. The function is limited to 8-bit stations (DSS-9756, DSS-9758, DSS-9763) with Arabic, Persian or Arabic-French firmware.

The following code variants are possible:

EBCDIC.EHC.LA.ARA  
 EBCDIC.EHC.LA.IND  
 EBCDIC.EHC.LF.INT  
 EBCDIC.EHC.LF.FAR  
 EBCDIC.EHC.NA.ARA  
 EBCDIC.EHC.NA.IND

You can only use Arabic/Persian 8-bit data display terminals if you have activated support for these data display terminals using the installation procedure. The installation procedure is described in the “VTSU” manual [1].

From the user’s point of view, Arabic/Persian 8-bit data display terminals are treated like European 8-bit data display terminals in XHCS. 8-bit mode can be set directly by programs using the VTSU control block or by activating the 8-bit user standard using the TIAM command MODIFY-TERMINAL-OPTIONS or via the VTSU operating parameters.

Names of Arabic/Persian 8-bit codes:

ISO code variant	XHCS name (CCSN)	Complete name	Display
6	EDF046 EEHCLAA	EBCDIC.DF.04-6.ARA EBCDIC.EHC.LA.ARA	Latin/Arabic with Arabic digits
A	EDF04A EEHCLAI	EBCDIC.DF.04-6.IND EBCDIC.EHC.LA.IND	Latin/Arabic with Indian digits
C	EDF04C EEHCLFI	EBCDIC.DF.04-FAR.INT EBCDIC.EHC.LF.INT	Latin/Farsi with international digits
D	EDF04D EEHCLFF	EBCDIC.DF.04-FAR.FAR EBCDIC.EHC.LF.FAR	Latin/Farsi with Persian digits
E	EDF04E EEHCNAA	EBCDIC.DF.04-NAF.ARA EBCDIC.EHC.NA.ARA	French/Arabic with Arabic digits
F	EDF04F EEHCNAI	EBCDIC.DF.04-NAF.IND EBCDIC.EHC.NA.IND	French/Arabic with Indian digits

A complete list of all valid CCSN (XHCS names) is provided starting on [page 143](#).

### Terminal emulation MT9750 in Unicode mode

The terminal emulation MT9750 V7.0 permits Unicode data to be sent to BS2000/OSD. DSS9763 must be set as the terminal type. To permit this, the session parameters of the emulation must be configured appropriately. This notifies VTSU that the terminal supports not only the EBCDIC character sets, but also the UTFE (Unicode) character set. This is required to enable the BS2000 command /MODIFY-TERMINAL-OPTIONS CCS = UTFE to be used.

When an application wants to send Unicode data to the emulation or receive Unicode data from it, VTSU must be notified of this. This can be done in the applications by means of the VTSU-B control block or using the BS2000 command /MODIFY-TERMINAL-OPTIONS. The setting in the VTSU-B control block has priority over the value set with /MODIFY-TERMINAL-OPTIONS. However, it only applies for the current call. For detailed information on configuring the terminal emulation MT9750, please refer to the overview manual "Unicode in BS2000/OSD" [\[32\]](#).

### 3.3.2 XHCS program interfaces

The XHCS subsystem provides for central management of code tables. To use the benefits of XHCS, various program interfaces are available.

- NLSCOD program interface - code information
  - Supplies and prepares different tables (one table per call).
  - The possible tables are:
    - translation table to another code
    - conversion table of lowercase to uppercase letters
    - table of sorting priorities for each character
    - table that combines a user-specified set of up to eight properties for each code character
    - table containing the highest Unicode position supported by XHCS, the size of the sorting information for Unicode, and the size of the normalization table
- NLSCNV program interface - character string conversion
  - Carries out the code conversion of character strings. A character string is converted from one CCS to another.
  - Outputs the length of Unicode strings
  - Converts lowercase letters to uppercase and vice versa
  - Normalizes input strings for Unicode
- NLSCMP program interface - compatible codes
  - Provides information regarding the compatibility of codes and the possibility of conversion.
- BS2000 command /ADD-CODE-TABLES - adding new code tybles dynamically
  - Permits dynamic modification and extension of existing code during ongoing operation (the command description starts on [page 132](#)).
- Other interfaces
  - The coded character set of a file (PLAM library elements) is made available by various DMS interfaces (ILAM).

A description of NLSCOD, NLSCNV and NLSCMP starts on [page 64](#).

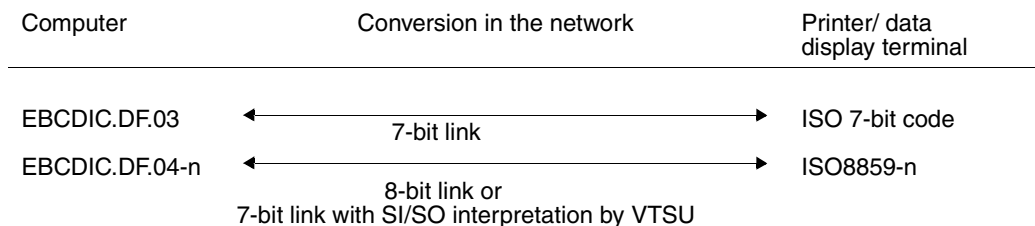
### 3.4 Code conversion

BS2000 computers work with an EBCDIC code but data display terminals and printers only process ISO codes. Consequently, when data is transferred between the computer and these peripherals, the ISO code must be converted to EBCDIC or vice versa.

The PDN (program system for remote data processing and network control) network software handles the conversion between these codes. Code conversion is dependent on the PDN generation, which in turn depends on the hardware attributes of the link and on the equipment connected (see the manuals for the data display terminals and the printers). An 8-bit data display terminal or 8-bit printer can be connected to the computer via either a 7- or 8-bit link (e.g. HDLC).

If an extended character set is transferred along a 7-bit link, a code extension mechanism is required. This involves using the control characters Shift In (SI) and Shift Out (SO), which are then interpreted by VTSU. These control characters (SI/SO) can double the message length.

The following figure illustrates code conversion between the computer and peripherals.



EBCDIC.DF.04-n is an extension of EBCDIC.DF.03-IRV (=International Reference Version). EBCDIC.DF.03-DRV (=Deutsche Referenz-Version/German reference version) is a variant of EBCDIC.DF.03-IRV. Both codes support character sets of the same size. They have a common character set, the EBCDIC kernel, and differ only in certain symbols (see the EBCDIC.DF.03 table). System and application programs not supported by XHCS use these EBCDIC.DF.03 codes.

ISO 8859-n is an extension of the ASCII code. ASCII is the US variant of the 7-bit ISO646 code. In addition to the international ASCII code, there are other national variants of the 7-bit ISO646 code. This 7-bit code is used by 7-bit data display terminals and 8-bit data display terminals in 7-bit mode. For a 7-bit data display terminal to interpret text or a keystroke correctly, it must be set to the correct mode (e.g. "International") locally via the keyboard (see SIDATA menus). The ISO 8859 extended character set has several country-specific variants.

The figures on the following pages illustrate examples of code conversion between the computer and peripherals.

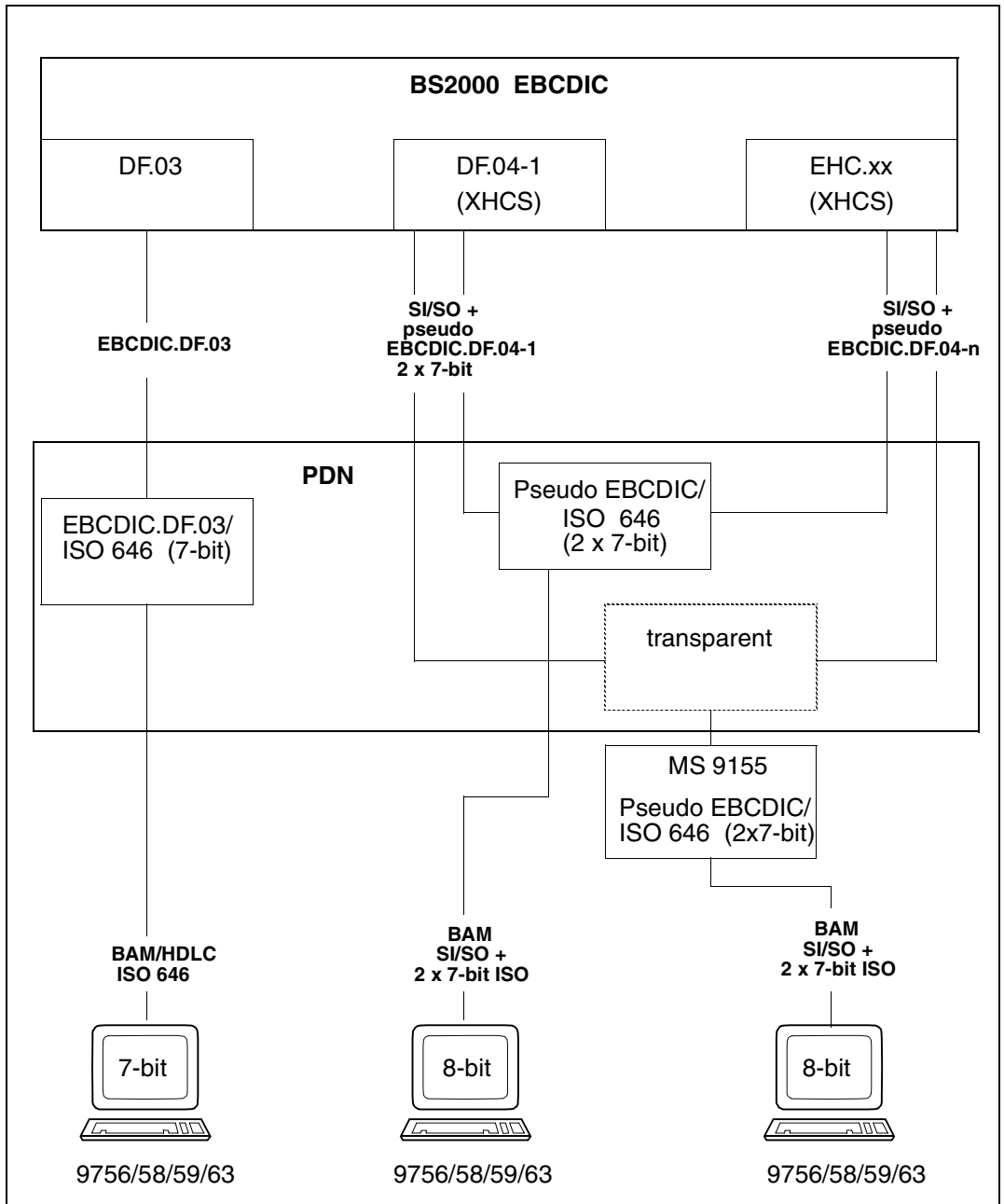


Figure 2: Code handling with XHCS in the BS2000 environment



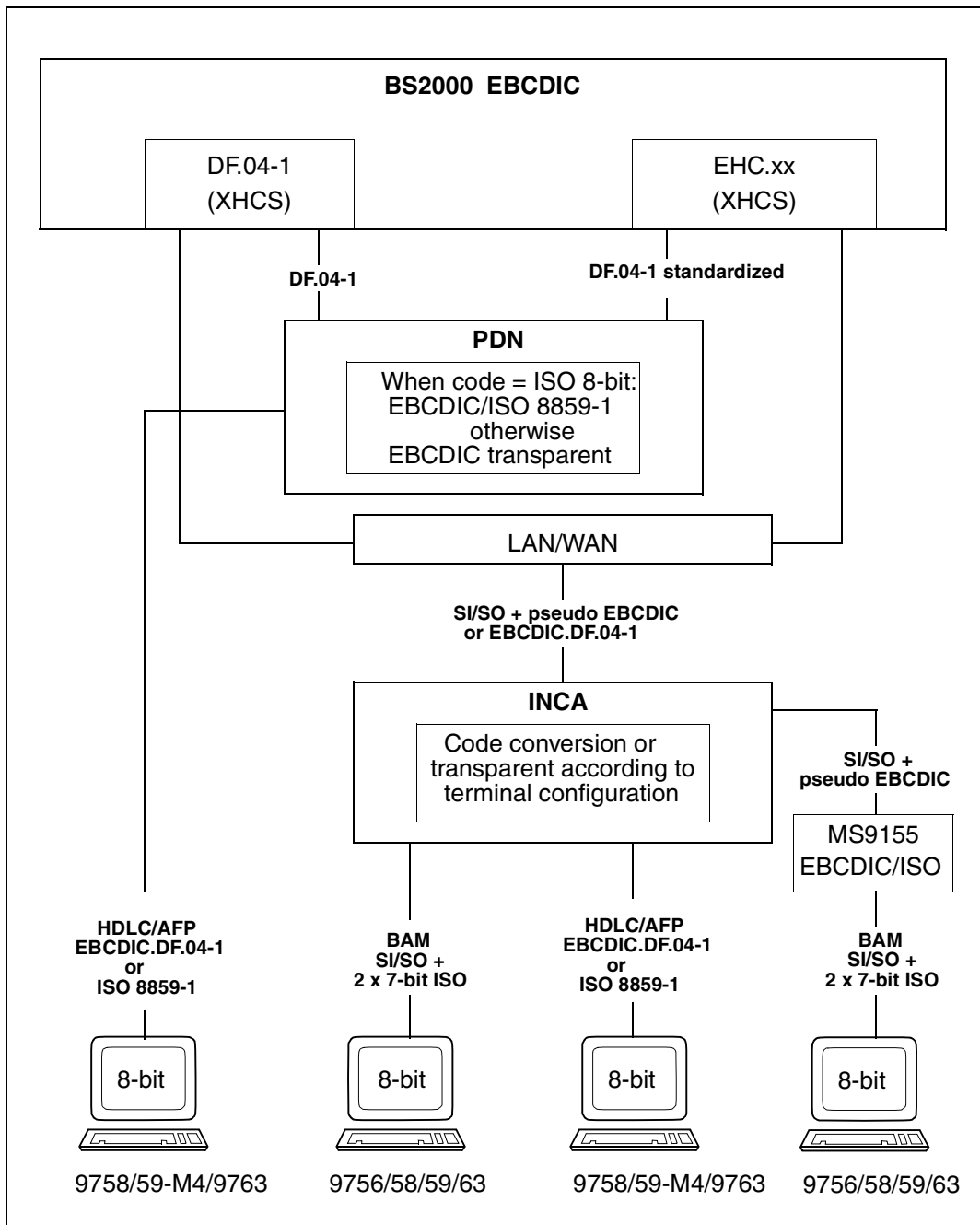


Figure 3: Code handling with XHCS in the BS2000 environment

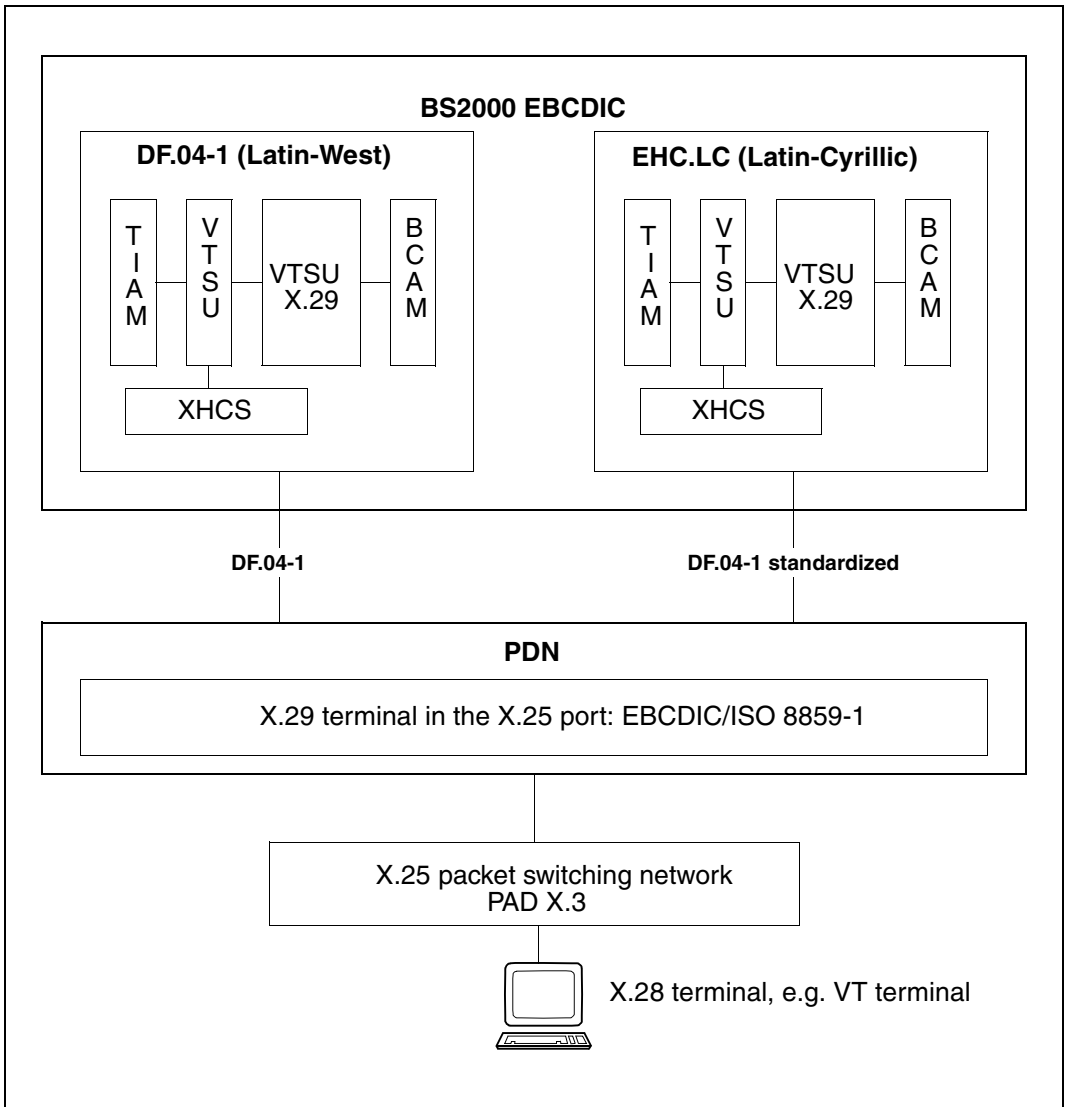


Figure 4: Code handling in BS2000 with VTSU - X.29 and XHCS in the BS2000 environment

The following diagram shows Unicode in the BS2000/OSD system environment in the form of the Unicode encodings on the individual interfaces. For the sake of greater clarity, some of the technical details have been omitted. Detailed information on this subject is provided in the manual “Unicode in BS2000/OSD” [32].

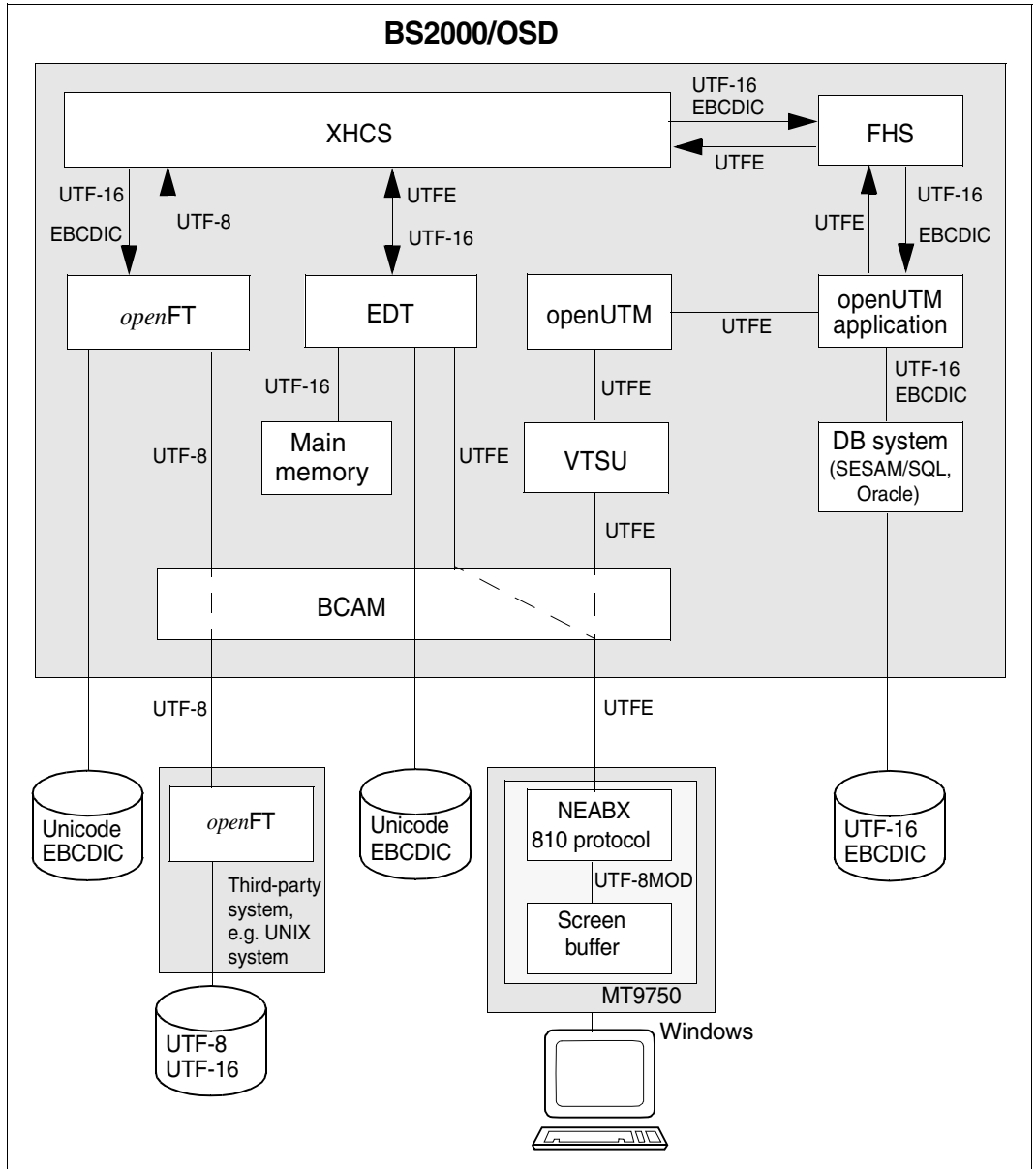


Figure 5: Overview of the affected interfaces by UNICODE

## 3.5 Using extended codes

The following section describes what you should take into consideration when using extended codes and provides you with information and tips for working with XHCS.

The following conditions must be met to ensure that programs will run correctly without being permanently linked to a predefined character set:

- 8-bit compatible equipment
- extended terminal protocol and information regarding the link
- identification of file and library element codes
- 8-bit mode control
- display of the computer code preferred by the user
- central code tables that contain conversion tables and additional information tables
- 8-bit codes, alphabet direction from left to right

### 3.5.1 Extended terminal protocols

The transmission protocol between the computer and terminal is extended in new equipment. The device mode (7-bit or 8-bit) can be dynamically set, and the character set can be specified by the application.

Since some terminals support different character sets, terminals can be queried as to whether they are 8-bit-capable or Unicode-capable and, if so, which character sets are available.

The terminal emulation MT9750 is Unicode-capable as of V7.0. However, the emulation's session parameters must then be configured appropriately.

You can obtain information about 8-bit capability and the supported character sets using the "terminal status" function, which depends on the access methods used (TSTAT for TIAM, YINQUIRE for DCAM). You can use the VTSU macro DCSTA (TYPE=BASIC) to obtain information regarding the relevant characteristics of the terminal and link.

The following DSTA macro fields relate to XHCS:

STATTYPE	Indicates whether the terminal can operate in 8-bit mode
STACCSNN	Number of supported 8-bit/Unicode character sets
STACSS1-16	Numbers of the supported 8-bit/Unicode character sets
STACURCH	Contains the name of the user default character set if the terminal is 8-bit capable and the character set is compatible with the terminal.
STAACTCH	Contains the name of the active extended character set.

These interfaces are also available for COBOL, C, PL/1 and FORTRAN.

## 3.5.2 Identification of file and library element codes

Since multiple codes can be used simultaneously in an existing system, XHCS identifies files and PLAM library elements with the CCSN (name of the coded character set). The program that reads or writes the data can use this information.

The CCSNs of the files are stored as a file attribute called CODED-CHARACTER-SET. The `MODIFY-FILE-ATTRIBUTES` command can be used to set this attribute, e.g.

```
/MODIFY-FILE-ATTR FILE-NAME=MYFILE,C-C-S=EDF041
```

The `SHOW-FILE-ATTRIBUTE` command returns the CCSNs of the specified files, e.g.

```
/SHOW-FILE-ATTRIBUTE FILE-NAME=MYFILE,INFO=PAR(ORG=YES)
```

## 3.5.3 Activating 8-bit mode/Unicode mode

8-bit mode or Unicode mode has to be set. This prevents you from accidentally generating 8-bit data or Unicode data which would then create problems when it is returned to 7-bit terminals.

A detailed description of Unicode is provided in the manual “Unicode in BS2000/OSD” [32].

The following tips apply only to 8-bit-capable or Unicode-capable terminals and assume that the desired CCS (character set) can be displayed. Attempts to activate a CCS that is not supported by the device are processed in 7-bit mode.

### 3.5.3.1 Explicit activation via the program

Programs that can process 8-bit files or Unicode files are able to set 8-bit mode or Unicode mode themselves. Since files and library elements have an attribute that indicates the code, these components can determine whether or not 8-bit files or Unicode files can be displayed on the terminal.

Explicit activation occurs by means of the VTSU control block (only with TIAM and DCAM).

### 3.5.3.2 Implicit activation via user commands

It is possible to work in 8-bit mode or Unicode mode even if the application does not request it. Users can specify that VTSU use an 8-bit code or Unicode code for its inputs and outputs. This is done by using the following TIAM command:

```
/MODIFY-TERMINAL-OPTIONS CODED-CHARACTER-SET=<ccs name of an 8-bit character set>  
or  
/MODIFY-TERMINAL-OPTIONS CODED-CHARACTER-SET=UTFE
```

This activates 8-bit mode with the CCS of the user default character set on the computer side and generates a “permanent 8-bit mode.”



In some cases, this command must be specified before the component that requires XHCS is called.

### 3.5.3.3 Automatic activation via the system configuration

A third option is less flexible since it applies to the entire BS2000 system. VTSU can be configured so that the user default character set is used with TIAM, openUTM and/or DCAM applications that are connected to an 8-bit terminal. This generates a “permanent 8-bit mode.”

## 3.5.4 User default character set

A user default character set can be assigned to each user. This is the same CCS that is used when 8-bit mode is activated and a code is not explicitly requested. It can be assigned by the system administrator using the ADD-USER or MODIFY-USER command. The SHOW-USER-ATTRIBUTES command is used to return the user default character set (refer to the manuals “BS2000/OSD-BC Commands, Volumes 1 - 5” [4]).

## 3.5.5 Centralizing code tables

Programs do not need to permanently store information regarding character sets. They receive information from the XHCS subsystem using the character set name (CCSN).

A code can only be used if it is defined in XHCS; this is the responsibility of the systems support staff. There is no direct way to tell which codes are available in the system. You can use EDT V16.4 or later to obtain this information indirectly; the @SHOW CCS statement displays a list of available character sets.

### 3.5.6 Applications with XHCS support

Existing programs (except for openUTM applications) can work only with XHCS support if the command

```
/MODIFY-TERMINAL-OPTIONS CODED-CHARACTER-SET= 8-BIT-DEFAULT
```

is entered before the program is started. If the command is not entered, the program works with the default code for 7-bit devices. The computer code is then EBCDIC.DF.03.

#### Code transparency

Programs that use XHCS must be independent in terms of the character set. Code transparency means that a program does not make any assumptions about the data code structure. This information (character classification, comparison sequences, information about uppercase letters) must be acquired from an external mechanism.

## 3.6 XHCS with FHS and IFG

In IFG V8.3 and later, you can generate and update formats for 8-bit terminals (9763 and 9758) or terminals in Unicode mode (MT9750). To do this, you must use 8-bit terminals or terminals in Unicode mode. Formats that were created exclusively on terminals in 8-bit mode are called 8-bit formats. Formats that were created on terminals in Unicode mode are called Unicode formats. You can convert formats of older IFG versions to 8-bit formats by changing the corresponding terminal group. The character set name (CCSN) of a format is later used by FHS to obtain the necessary code tables for determining the characters that can be displayed or the conversion of lowercase to uppercase letters.

By changing the terminal group, you can also generate formats for 8-bit printers.

For information on how you can use 8-bit support or Unicode support in format mode, refer to the relevant “FHS” [21] and “IFG” [22] manuals.

### 3.6.1 TIAM applications

A TIAM application can send 8-bit messages or Unicode messages only to 8-bit terminals or terminals in Unicode mode (generally a terminal emulation). The status of the terminal is queried automatically when the connection is established. When you receive the status response, you can use the TSTAT macro to request information on the data display terminal status (e.g. whether it has 8-bit capability).

The TSTAT macro provides the following information:

- The terminal type. The terminal is either a 7-bit terminal, an 8-bit terminal or a terminal in Unicode mode.
- The name of the extended standard code when it is possible to work in 8-bit mode.
- The name of the Unicode when it is possible to work in Unicode mode.
- The name of the extended standard code activated if an 8-bit mode is switched on (MODIFY-TERMINAL-OPTIONS command)
- The name of the Unicode activated if a Unicode mode is switched on (MODIFY-TERMINAL-OPTIONS command).
- A list of the ISO code variants supported.

During LOGON processing a check is then made as to whether the standard code used is compatible with the codes supported by the current terminal. If the code is not supported, a warning is issued and a 7-bit link is assumed.



The CCSNAME parameter in the VTSU control block allows you to set 8-bit mode or Unicode mode for a TIAM application. Unless you do this, the data display terminal automatically works in the default mode (7-bit mode or an 8-bit mode activated using the MODIFY-TERMINAL-OPTIONS command). This guarantees full data compatibility.

You can specify one of two values for the CCSNAME parameter in the VTSU control block:

- When CCSNAME=\*EXTEND, the extended standard code is used automatically.
- When CCSNAME=ccsname, the name of the character set to be used is specified explicitly. This can be EDF03IRV, the name of an extended standard code or a Unicode.

You assign the code to the appropriate format yourself. The system does not check this assignment. If you assign an incompatible code to the format, the following errors can occur:

- If an 8-bit code or Unicode is assigned in the VTSU control block and a 7-bit format is sent, a formatting error can occur because FHS does not recognize the characters of an extended character set.
- If a 7-bit code is assigned in the VTSU control block and an 8-bit format or Unicode format is sent, an output error can occur because VTSU does not recognize the characters of an extended character set or of a Unicode character set.
- If an 8-bit format is sent and another 8-bit code is assigned in the VTSU control block, errors can occur because FHS uses a different code for formatting to that used by VTSU for output editing.
- If a Unicode format is sent and an 8-bit code is defined in the VTSU control block, errors can occur because FHS uses a different code for formatting to that used by VTSU for output editing.

A TIAM application which uses 8-bit formats or Unicode operates correctly if the CCS of the format agrees with the CCS which VTSU uses.

### 3.6.2 DCAM applications

A DCAM application can send 8-bit messages or Unicode messages only to 8-bit terminals, to terminals in Unicode mode and to 8-bit printers. To ascertain the data display terminal type when using a DCAM application, you must request the status of the data display terminal yourself (PROC = TERMSTAT in the CCB for a YOPNCON call). When you receive the status response, you can use the YINQUIRE macro to request information on the data display terminal status (e.g. whether it has 8-bit capability). The YINQUIRE macro provides the following information:

- The terminal type. The terminal is either a 7-bit terminal, an 8-bit terminal or a terminal in Unicode mode.
- The name of the extended standard code when it is possible to work in 8-bit mode.
- The name of the Unicode when it is possible to work in Unicode mode.
- A list of the ISO code variants supported.

On connection setup, a check is made as to whether the standard code used is compatible with the codes supported by the current terminal. If the code is not supported, a 7-bit link is assumed; no warning is issued.

The CCSNAME parameter in the VTSU control block allows you to set 8-bit mode for a TIAM application. Unless you do this, the data display terminal works in 7-bit mode automatically. This guarantees full data compatibility.

You can specify one of two values for the CCSNAME parameter in the VTSU control block:

- When CCSNAME=\*EXTEND, the extended standard code is used automatically.
- When CCSNAME=ccsname, the name of the character set to be used is specified explicitly. This can be EDF03IRV, the name of an extended standard code or a Unicode.

You assign the code to the appropriate format yourself. The system does not check this assignment. If you assign an incompatible code to the format, the following errors can occur:

- If an 8-bit code or a Unicode is defined in the VTSU control block and a 7-bit format is sent, a formatting error can occur because FHS does not recognize the characters of an extended character set.
- If a 7-bit code is defined in the VTSU control block and an 8-bit format or Unicode format is sent, an output error can occur because VTSU does not recognize the characters of an extended character set or of a Unicode character set.
- If an 8-bit code format is sent and another 8-bit code is defined in the VTSU control block, errors can occur because FHS uses a different code for formatting to that used by VTSU for output editing.

- If a Unicode code format is sent and an 8-bit code is defined in the VTSU control block, errors can occur because FHS uses a different code for formatting to that used by VTSU for output editing.

### 3.6.3 openUTM applications

In openUTM applications the CCSNAME parameter can be generated directly for each USER and for each LTERM partner. The specified CCS name must belong to an EBCDIC character set which is defined in the BS2000 system or to the Unicode variant UTFE. If the terminal or terminal emulation (e.g. MT9750) is 8-bit-capable, 8-bit mode is supported. If the format sent by an openUTM application is an 8-bit format or a Unicode format and if the standard code used is compatible with the codes which are supported by the current terminal, the 8-bit terminal or emulation automatically works in 8-bit mode or Unicode mode. Compatibility is checked on connection setup. If the code is not compatible, a 7-bit link is assumed; no warning is issued.

When generating the openUTM application, note that 8-bit terminals or terminals in Unicode mode must be generated appropriately (PTERM statement in KDCDEF). The KDCFILE declaration must agree with the PDN generation.

The extended standard code or the Unicode used for format preparation must be identical to the standard code or the Unicode of the openUTM application. If it is not, errors can occur because FHS uses a different code for formatting to that used by VTSU for output editing.

## 3.7 XHCS without FHS and IFG

The sections that follow describe the use of XHCS in TIAM, DCAM and openUTM applications without FHS and IFG.

### 3.7.1 TIAM applications

A TIAM application can send 8-bit messages or Unicode messages only to 8-bit terminals or terminals in Unicode mode. The status of the terminal is queried automatically when the connection is established. When you receive the status response, you can use the TSTAT macro to request information on the status of the data display terminal (e.g. whether it has 8-bit capability). The TSTAT macro supplies the following information:

- The terminal type. The terminal is either a 7-bit terminal, an 8-bit terminal or a terminal in Unicode mode.
- The name of the extended standard code when it is possible to work in 8-bit mode.
- The name of the extended standard code activated if an 8-bit mode is switched on (MODIFY-TERMINAL-OPTIONS command)
- The name of the Unicode activated when Unicode mode is switched on (MODIFY-TERMINAL-OPTIONS command).
- A list of the ISO code variants supported.

During LOGON processing a check is then made as to whether the standard code used is compatible with the codes supported by the current terminal. If the code is not supported, a warning is issued and a 7-bit link is assumed.

The CCSNAME parameter in the VTSU control block allows you to set 8-bit mode or Unicode mode for a TIAM application. Unless you do this, the data display terminal automatically works in the default mode (7-bit mode or an 8-bit mode activated using the MODIFY-TERMINAL-OPTIONS command). This guarantees full data compatibility.

You can specify one of two values for the CCSNAME parameter in the VTSU control block:

- When CCSNAME=\*EXTEND, the extended standard code is used automatically.
- When CCSNAME=ccsname, the name of the character set to be used is specified explicitly. This can be EDF03IRV, the name of an extended standard code or a Unicode.

The CCSNAME parameter can be used in extended line mode and in physical mode. In physical mode you must ensure that the fields required by VTSU (e.g. ISOA, PAR01L) are available in the message header.

### 3.7.2 DCAM applications

A DCAM application can send 8-bit messages or Unicode messages only to 8-bit terminals and, to terminals in Unicode mode and to 8-bit printers. To ascertain the terminal type, you must request the status of the data display terminal (PROC = TERMSTAT in the CCB for a YOPNCON call). When you receive the status response, you can use the YINQUIRE macro to request information on the data display terminal status (e.g. whether it has 8-bit capability).

The YINQUIRE macro supplies the following information:

- The terminal type. The terminal is either a 7-bit terminal, an 8-bit terminal or a terminal in Unicode mode.
- The name of the extended standard code when it is possible to work in 8-bit mode.
- The name of the Unicode when it is possible to work in Unicode mode.
- A list of the ISO code variants supported.

On connection setup, a check is made as to whether the standard code used is compatible with the codes supported by the current terminal. If the code is not supported, a 7-bit link is assumed; no warning is issued.

The CCSNAME parameter in the VTSU control block allows you to set 8-bit mode or Unicode mode for a DCAM application. Unless you do this, the data display terminal works in 7-bit mode automatically. This guarantees full data compatibility.

You can specify one of two values for the CCSNAME parameter in the VTSU control block:

- When CCSNAME=\*EXTEND, the extended standard code is used automatically.
- When CCSNAME=ccsname, the name of the character set to be used is specified explicitly. This can be EDF03IRV, the name of an extended standard code or a Unicode.

The CCSNAME parameter can be used in extended line mode and in physical mode. In physical mode you must ensure that the fields required by VTSU (e.g. ISOA, PAR01L) are available in the message header.

### 3.7.3 openUTM applications

In openUTM applications the CCSNAME parameter can be generated directly for each USER and for each LTERM partner. The specified CCS name must belong to an EBCDIC character set which is defined in the BS2000 system or to the Unicode variant UTFE. If the terminal or terminal emulation (e.g. MT9750) is 8-bit-capable, 8-bit mode is supported.

### 3.7.4 Recommendations

This section contains a few guidelines for programmers of TIAM or DCAM applications.

#### 3.7.4.1 Determining the name of the data character set (CCSN)

If the input to be read is a file or a PLAM library element, the CCSN is returned by different DMS or ILAM interfaces. If this type of input does not exist, the name of the computer code used must be determined.

With terminal inputs and outputs, you receive the CCSN used depending on the data type of the input or output:

- Tasks that process interactive terminal inputs/outputs (WRTRD) receive the CCSN from the return value of the TIAM TSTAT macro (STAACTCH field).
- Tasks that read their input from the SYSDTA system file receive the CCSN from SYSDTA GCCSN (CCSNCCSN field).

#### 3.7.4.2 Availability of XHCS

XHCS-SYS (the subsystem of XHCS) is not a standard component of BS2000. You should always check whether it is available before carrying out your operation. To prevent inconsistent responses from components that call XHCS, the XHCS subsystem cannot be deleted or unloaded.

#### 3.7.4.3 Detecting ISO codes

The type of code (EBCDIC or ISO/ASCII) is detected by XHCS (NLSCMP interface). ISO/ASCII codes should be rejected since BS2000 is an EBCDIC-oriented system.

#### 3.7.4.4 Terminal capabilities

When processing 8-bit terminal inputs/outputs, you should ensure that the desired character set can be displayed on the terminal (8-bit capability and support of the relevant ISO variant).

The TSTAT (TIAM) and YINQUIRE (DCAM) macros provide information (DCSTA) on 8-bit capability (STATYPE field) and supported codes in the form of a list of supported ISO variants. All 8-bit codes whose ISO variants are included in this list can be used. The number of the ISO variant of a code is returned by the NLSCMP macro to XHCS.

### 3.7.4.5 Complete and restricted codes

VTSU supports the following codes:

- In 7-bit mode: supports the international version of EBCDIC.DF.03 (does not support XHCS services, i.e. with internal tables).
- In 8-bit mode: supports 8-bit codes that are defined in the XHCS tables and are compatible with the ISO 8859 variant supported by the terminal.
- In Unicode mode: supports Unicodes codes that are defined in the XHCS tables and are compatible with the ISO 8859 variant supported by the terminal.

Data whose CCSN is blank or EDF03IRV has to be edited in 7-bit mode.

To determine whether an existing code name identifies a 7- or 8-bit code, use the NLSCMP interface to call XHCS.

### 3.7.4.6 Working in 8-bit mode

- Terminal inputs/outputs

The VTSU control block (VTSUCB) specifies the CCSN for VTSU. If you work in 8-bit mode on an 8-bit terminal, the CCSN value in VTSUCB must be set either to the name of an 8-bit code that is supported by the terminal or to blank when the following command is used:

```
/MODIFY-TERMINAL-OPTIONS CODED-CHARACTER-SET=8-BIT-DEFAULT
```

- CCSN of statements

The SDF macro RDSTMT should be run with the CCSNAME that is returned by the SYSDATA macro GCCSN. Statements that are read in 7-bit mode are not incompatible with processing in 8-bit mode.

- CCSN of output files

The appropriate CCSN is explicitly used as a file/element attribute. The CCSN attribute of SYSOUT and SYSLST can be inquired by means of the SYSDATA macro GCCSN.

- Other information about codes

Information such as lowercase to uppercase letter conversion, character properties or sort sequences is provided by XHCS when the NLSCOD macro is called.

### 3.7.4.7 Working in Unicode mode

- Terminal inputs/outputs  
The VTSU control block (VTSUCB) specifies the CCSN for VTSU. If you work in Unicode mode on an 8-bit terminal, the CCSN value in VTSUCB must be set either to the name of a Unicode that is supported by the terminal or to blank when the following command is used:  

```
/MODIFY-TERMINAL-OPTIONS CODED-CHARACTER-SET=UTFE
```
- CCSN of statements  
The SDF macro RDSTMT should be run with the CCSNAME that is returned by the SYSDFILE macro GCCSN. Statements that are read in 7-bit mode are not incompatible with processing in Unicode mode.
- CCSN of output files  
The appropriate CCSN is explicitly used as a file/element attribute. The CCSN attribute of SYSOUT and SYSLST can be inquired by means of the SYSDFILE macro GCCSN.
- Other information about codes  
Information such as lowercase to uppercase letter conversion, character properties or sort sequences is provided by XHCS when the NLSCOD macro is called.

### 3.7.4.8 Working in 7-bit mode

- Terminal inputs/outputs  
If you work in 7-bit mode on an 8-bit terminal, the CCSN value in VTSUCB must be set to “EDF03IRV”. This value ensures that processing will be carried out in 7-bit mode by VTSU using its own tables. The only 7-bit code for terminal inputs and outputs is EDF03IRV; other 7-bit codes are rejected by VTSU.
- CCSN of output files  
The CCSN of files or PLAM library elements must be set to blank if its contents has to be interpreted in a 7-bit context.
- Other information about codes  
Whenever you work in 7-bit mode, use internal tables. This has the advantage that your work is independent of XHCS (availability, modification of tables) and full compatibility to older application versions is ensured. However, if 7-bit codes are required, you can use the XHCS tables for 7-bit codes.



## 3.8 System software

The following section explains which system software versions are XHCS capable and what you may need to take into consideration.

### 3.8.1 EDT V17.0

EDT can generate files in all character sets which are supported by XHCS.

The code used can be modified explicitly (@CODENAME statement) or implicitly (if the CCSN of the file that is read in is used).

An active EDT application does not need to have a homogeneous code environment set. EDT always performs code conversion and permits different character sets.

If extended characters are not supported by the terminal, these are displayed using smudge characters, e.g. when 8-bit files are displayed on a 7-bit terminal.

### 3.8.2 SHOW-FILE

Files or PLAM library elements are correctly displayed by the SHOW-FILE command if their CCSN attribute identifies an 8-bit code supported by the terminal. In such cases, extended characters can also be entered in a search string (FIND statement).

If extended characters are not supported by the terminal, these are displayed using smudge characters (Unicode only in BS2000/OSD V7 or higher), e.g. when 8-bit files are displayed on a 7-bit terminal.

### 3.8.3 IFG V8.3

IFG can be used to define 8-bit CCSs or Unicode CCs that are used for creating a format. This information is stored with the format and can subsequently be used by FHS for input/output formatting.

8-bit formats can only be generated, displayed, and modified on an 8-bit terminal; the terminal group specified in the user profile must match the 8-bit terminal.

Unicode formats can only be generated, displayed, and modified on a terminal in Unicode mode; the terminal group selected in the user profile must also match the terminal in Unicode mode.

Some restrictions for 8-bit formats or Unicode formats do exist: rapid formatting cannot be selected and the formats cannot be used with ICE.

### 3.8.4 FHS V8.3

The input/output formatting carried out by FHS requires various tables (individual character assignment, lowercase to uppercase conversion).

FHS can use up to three different table sets:

- the user-defined table set (in the MFHSCTAB module)
- the XHCS table set
- the FHS standard table set

Unlike the user-defined or XHCS table sets, the FHS standard table set is always available.

Of these three table sets, the user-defined table set has the highest priority. This means that if a user-defined table set exists, it will be used for processing. The XHCS table set is used only if a user-defined table set does not exist. If neither a user-defined table set nor an XHCS table set exists, the standard table set is used.

### 3.8.5 PERCON

PERCON V2.5 and higher offers XHCS support, and V2.9 and higher Unicode support.

PERCON is the BS2000 program for transferring and converting files. Extended codes are supported by PERCON. During processing, PERCON takes the statement, input and output CCS into consideration. PERCON also implements conversion if:

- The input CCS is different from the output CCS. However, input and output CCSs must be compatible.
- The statement CCS is different from the input CCS. If c-strings are used in the SELECT-INPUT-RECORDS statement, statement and input CCSs must belong to the same code family and all characters of the c-strings must be present in the input CCS.
- The statement CCS is different from the output CCS. If c-strings are used in the SET-GROUP-ATTRIBUTES, SET-RECORD-MAPPING- or SET-PAGE-LAYOUT statement, statement and output CCSs must belong to the same code family and all characters of the c-strings must be present in the output CCS.

In addition, sorted files can be checked for ascending or descending order independently of the CCS.

In addition to conversion, PERCON V2.9 and higher can also use the normalization function for UTF-16.

### 3.8.6 SORT

SORT V7.4 and higher offers XHCS support, and V7.9 and higher Unicode support.

SORT can base its processing on the sort sequences encoded in the XHCS tables. This means that sorting is not based on the character display or on SORT-internal data. The requirements for this are:

- the EXTENDED-CHARACTER format must be selected (in the SORT-RECORDS statement)
- files to be sorted have a CCSN that identifies an existing 8-bit CCS.

Please note that the XHCS sorting priority tables (in which each character is assigned a position in the sort sequence) must be unique; each sorting priority appears only once in a table. This means that no two characters can have the same value in the sort process. The data CCS cannot be an 8-bit CCS.

As of SORT V7.9, SORT can use the sort table provided by XHCS for the Unicode variant UTF-16. To permit this, you must select the UNICODE-CHARACTER format in the SORT statement.

### 3.8.7 RSO V3.5

To convert file contents, RSO uses the coded character set name of the file to be printed. RSO requests the relevant conversion table from XHCS, providing that the remote printer is defined as an “8-bit” type in the RSO configuration file and the CCSN is neither blank nor EDF03IRV. Please note that the selection of the printer character set and the CCSN file are two completely different aspects.

### 3.8.8 LMS V3.3B

LMS supports extended codes only on a limited basis.

Like files, library elements also have a CCSN attribute. If a file is transferred to a PLAM library, the CCSN of the file is maintained; in the other direction, a selected element generates a file with the CCSN of the element. The CCSN is also maintained when elements are duplicated.

Coded character set names can be explicitly set using the `MODIFY-ELEMENT-ATTRIBUTES` statement. You can use the operands `SHOW-ELEMENT-ATTRIBUTES` and `INFORMATION=MAXIMUM` to view the CCSN of an element.

### 3.8.9 FT-BS2000 V5.0

If files are transferred between two BS2000 systems, the CCSN attribute of FT-BS2000 is maintained. For exchanging text files with FTAM partners, FT-BS2000 uses the EBCDIC.DF.04/ISO 8859 conversion table to convert the characters.

FT-BS2000 checks whether the transferred CCSN is present in the receiving system and compatible with the reference code. If this is the case, the receive data is converted in accordance with the code table. The CCSN attribute is stored in the receive file.

### 3.8.10 OMNIS V6.3

OMNIS as of V6.3 is required to work in 8-bit mode. If OMNIS is used to set up a connection to a remote processor, the following must be taken into consideration:

- the coded character sets used must be defined in both systems (same name, same contents)
- both VTSU versions (version 10.0 or later) must support XHCS
- permanent 8-bit mode is possible only with VTSU as of V11.0 on both sides.

### 3.8.11 Examples

*Example 1:* Converting a file from one character set to another

The following example shows how you can use the PERCON program to convert the contents of a file from one character set to another. The input file, in this example PERCON.IN, has a character set attribute that corresponds to its contents, e.g. EDF03IRV.

1. Create an empty output file with the desired CCS:

```
/CREATE-FILE PERCON.OUT,C-C-S=EDF041
/SET-FILE-LINK LINK-NAME=POUT,FILE-NAME=PERCON.OUT,ACCESS-METHOD=SAM
```

2. Call PERCON:

```
/START-PERCON
//ASS-INPUT-FILE FILE=DISK-FILE(NAME=PERCON.IN),LINK-NAME=PIN
    assigns the input file
//ASS-OUTPUT-FILE LINK-NAME=POUT
    assigns the output file
//END
```

starts the conversion and ends the program

*Example 2:* Sorting a file by means of the XHCS tables

As illustrated in the following example, the SORT program can use the XHCS tables for sorting:

```
/START-SORT
    starts the SORT program
//ASSIGN-FILES INPUT-FILE=SORT.8.IN,OUTPUT-FILE=SORT.8.OUT
    assigns the input and output file
//SORT-RECORDS FIELDS=FIELD-EXPLICIT(5,10,FORMAT=EXTENDED-CHARACTER)
    the use of XHCS tables must be explicitly specified using the Format parameter
//END
```



---

## 4 XHCS macros

XHCS provides three ASSEMBLER macros with the following functions:

- The NLSCOD macro provides various information on the source character set. The information provided depends on the TABLE parameter. Depending on the value of the TABLE parameter, you are provided with a table for conversion between code sets, for converting lowercase letters to uppercase and vice versa, or for specifying the collating sequence of characters or character properties. A byte properties table can be output for UTFE.
- The NLSCMP macro provides information on code compatibility. The information provided depends on the specified ISO code variant number or code name. Depending on what you specify, you are provided with a list of compatible codes.
- The NLSCNV macro converts strings directly. Strings specified here are converted directly to a compatible target code. It also converts lowercase letters to uppercase letters and vice versa, outputs the length of Unicode strings, and normalizes input strings. Conversion which can be reversed unambiguously can be implemented using the appropriate wildcard character processing for 7-/8-bit character sets. In addition, it is possible to branch to the NLSCNV function from TU without an SVC.

At the NLSCNV and NLSCMP interfaces you can now not only specify and receive the input/output strings in V format, but also handle the strings using an address and a length.





<code>=S</code>	generates a parameter list. The fields are filled in accordance with the specified keyword parameters of the macro. Names and equates are not generated. A command component is generated. The standard header is set automatically. No commands are generated for interpreting the return code.
<code>=L</code>	generates a parameter list. The fields are filled in accordance with the specified keyword parameters of the macro. Names and equates are not generated. The standard header is set automatically.
<b>PREFIX</b>	
<code>=x</code>	specifies the first character that precedes names defined with <code>MF=D</code> or <code>MF=C</code> . The default value is <code>PREFIX=G</code> .
<b>MACID</b>	
<code>=xxx</code>	specifies the 2nd, 3rd and 4th characters of names defined with <code>MF=C</code> . The default value is <code>MACID=NLT</code> (also for <code>MF=D</code> ).
<b>PARAM</b>	
<code>=paramlist</code>	specifies the name of the parameter list; contains the
<code>=(register)</code>	address of the parameter list.
<b>CCSNAME</b>	
	specifies the name of the source code. The name is not more than 8 bytes in length. If you do not specify a name, EDF03IRV automatically becomes the source code.
	The only Unicode variant that can be specified is UNICODE or UTFE. It is not permissible to specify UTF16 and UTF8 at this interface.
<code>=&lt;7-/8-bit-code&gt;</code>	Name of the 7-/8-bit code.
<code>=UNICODE</code>	The Unicode character set UTF16 is the source code.
<code>=UTFE</code>	The Unicode character set UTFE is the source code.
<code>=*SYSDEF</code>	The system default code is the source code
<code>=*USRDEF</code>	The user default code is the source code.

TABLE	Table type to be supplied.
=CONVERT	<p>Table for converting the source code (CCSNAME) to the target code (TOCCS).</p> <p>CCSNAME = 7-/8-bit-code and TOCCS = 7-/8-bit-code: Outputs a 256-byte conversion table.</p> <p>CCSNAME = 7-/8-bit-code and TOCCS = UNICODE: A 2*256-byte table is output which assigns the corresponding value in the Unicode table to each code point in the 8-bit table.</p> <p>CCSNAME = UNICODE and TOCCS = 7-/8-bit-code: A 64K table is output which assigns the corresponding value in the 8-bit table specified to each code point in the Unicode table if this is defined, and 0 if it is not defined.</p> <p>CCSNAME = UNICODE und TOCCS = UNICODE: This combination is not permitted.</p> <p>CCSNAME=UTFE: This specification is not permitted.</p>
=TOUPPER	<p>Table for converting lowercase letters to the corresponding uppercase letters.</p> <p>CCSNAME =7-/8-bit-code: Each lowercase letter is assigned the corresponding uppercase letter. All other positions are supplied with the same mapping.</p> <p>CCSNAME = UNICODE: Each lowercase letter in the Unicode table is assigned the corresponding uppercase letter. All other Unicode positions are supplied with the same mapping The table output is 64K*2 in length.</p> <p>CCSNAME=UTFE: Each lowercase letter which is coded in 1 byte is assigned the corresponding uppercase letter. All other positions are supplied with the same mapping. The table output is 256 bytes in length.</p>

=TOLOWER	<p>Table for converting uppercase letters to the corresponding lowercase letters.</p> <p>CCSNAME = 7-/8-bit-code: Each uppercase letter is assigned the corresponding lowercase letter. All other positions are supplied with the same mapping.</p> <p>CCSNAME = UNICODE: Each uppercase letter in the Unicode table is assigned the corresponding lowercase letter. All other Unicode positions are supplied with the same mapping The table output is 64K*2 in length.</p> <p>CCSNAME=UTFE: Each uppercase letter which is coded in 1 byte is assigned the corresponding lowercase letter. All other positions are supplied with the same mapping. The table output is 256 bytes in length.</p>
=SORT	<p>Table that specifies the collating sequence. This table can be used with the SORT utility routine to sort fields (Format=EXTENDED-CHARACTER when not Unicode, Format=UNICODE-CHARACTER when CCSN=UNICODE) (see the "SORT (BS2000/OSD)" manual [28]).</p> <p>CCSNAME = 7-/8-bit-code: Outputs a 256-byte collation table for 7-/8-bit codes.</p> <p>CCSNAME = UNICODE: Information on the Unicode Collation Algorithm is available at <a href="http://www.unicode.org/reports/tr10/">http://www.unicode.org/reports/tr10/</a>. The part that corresponds to the characters supported by XHCS has been filtered out of the <a href="http://www.unicode.org/Public/UCA/4.0.0/allkeys-4.0.0.txt">http://www.unicode.org/Public/UCA/4.0.0/allkeys-4.0.0.txt</a> file, which represents a standard collation table.</p> <p>A collation element of length 8 is available for each code point up to the highest supported value. The code points not used include elements with the assignment X'00..00' .</p>

The various table entries have the following format (for more information on variable collation elements and the sorting priority, please refer to the “SORT V7.9A (BS2000/OSD)” manual [28]):

Byte 1-2	Most significant bit	1: Variable collation element 0: All other cases
	Remaining 15 bits	Sorting priority for level 1 of element 1
Byte 3-4	Byte 2	Sorting priority for level 2 of element 1
	Byte 3	Sorting priority for level 3 of element 1
Byte 5-6	Sorting priority for level 4 of element 1	
Byte 7-8	0 or reference to a 2nd collation element which may exist: number of the entry of the 2nd collation element. The distance of this element from the start of the table is obtained by multiplying this number by 8.	

The second collation element has the same format as the first; it can also refer to a third element. The third element has precisely the same format, but always has the value X'0000' in its pointer.

The table initially consists of X'3000'=12288 entries, each 8 bytes long, which corresponds to the maximum possible number of Unicode positions from 0 through 2FFF.

The number of an entry corresponds to its Unicode position. The entries are assigned the first collation element of a collation entry.

After this range of X'3000'\*8 = X'24000' bytes, the range of the 2nd and 3rd collation elements begins. This is X'2000' bytes long and consequently has space for 1024 (=X'2000' / 8) entries of 2nd and 3rd collation elements.

CCSNAME=UTFE:

This specification is not permitted.

**=INFORMATION**

**CCSNAME = 7-/8-bit-code:**

This specification is not permitted.

**CCSNAME = UNICODE:**

Outputs three words with the following information:

**1st word:**

Highest Unicode position supported by XHCS. All Unicode tables are empty after this position. The highest supported Unicode position is currently 12287 decimal (= 2FFF hexadecimal).

**2nd word:**

Size of the collation information for Unicode:  
currently 106496 decimal (= 1A000 hexadecimal).

**3rd word:**

Size of the normalization table:  
currently 2777736 decimal (= 2A6288 hexadecimal)

**CCSNAME=UTFE:**

This specification is not permitted.

=(prop1,...,prop8)

A table with properties is supplied.

The table is structured as follows: The *i*th bit of each byte in the table is set to binary 1 when the *i*th property in the list applies to the relevant character. Up to 8 properties can be specified in any order. The property furthest to the right is mapped to the bit with the lowest value. If fewer than 8 properties are specified, the bytes are filled with binary zeros from the left.

The following properties are possible:

LETTER: the character is a letter.

DIGIT: the character is a digit.

ARITH: the character is an arithmetic character.

SPECIAL: the character is a special character.

LOWER: the character is a lowercase letter.

KERNEL: the character belongs to the EBCDIC kernel.

DEFINED: the character is defined in the CODE.

DISPL: the character is displayable/printable.

TRUE: binary 1 is always supplied.

FALSE: binary 0 is always supplied.

CCSNAME = 7-/8-bit-code:

Outputs a 256-byte properties table.

CCSNAME = UNICODE:

The properties table for Unicode is defined in the table module GNLMTAB.

The table contains 1 byte for each code point up to the highest supported value.

The table output is 64K in length.

CCSNAME = UTFE:

This specification is not permitted.

*Example 1*

In this example, it is to be verified whether the properties DISPL, DEFINED, KERNEL and ARITH apply to a character. ARITH is to be mapped to the bit with the lowest value. The other properties (LETTER, SPECIAL, DIGIT, LOWER) of the character are not requested. The code is EBCDIC.DF.04-1.

Structure of the parameter list:

```
NLSCOD MF=L, TABLE=(DISPL, DEFINED, KERNEL, ARITH),
        CCSNAME=EDF041, TABADDR=PROPTAB
```

Result for the character '?' (X'6F'):

The property ARITH does not apply to this character.

Representation:

Bit	0	0	0	0	1	1	1	0
Prop- erties					DISPL	DEFINED	KERNEL	ARITH

*Example 2*

In this example, it is to be verified whether the properties DISPL, DEFINED, KERNEL and ARITH apply to a character. The other bits are set to 1.

Structure of the parameter list:

```
NLSCOD MF=L, TABLE=(DISPL, DEFINED, KERNEL,
        ARITH, TRUE, TRUE, TRUE, TRUE),
        CCSNAME=EDF041, TABADDR=PROPTAB
```

Result for the character '?' (X'6F'):

The property ARITH does not apply to this character.

Representation:

Bit	1	1	1	0	1	1	1	1
Pro- perties	DISPL	DEFINED	KERNEL	ARITH	Rem. bits			

=(<utfeprop1>, ..., <utfeprop8>)

Outputs a 256-byte table with the byte properties for UTFE.

The table is structured as follows:

The *i*th bit of each byte in the table is set to binary 1 when the *i*th property in the list applies to the relevant byte. Up to 8 properties can be specified in any order. The property furthest to the right is mapped to the bit with the lowest value. If fewer than 8 properties are specified, the bytes are filled with binary zeros from the left.

The following properties are possible:

UTFE1	The byte is the first byte in a one-byte encoding
UTFE2	The byte is the first byte in a two-byte encoding
UTFE3	The byte is the first byte in a three-byte encoding
UTFE4	The byte is the first byte in a four-byte encoding
UTFE5	The byte is the first byte in a five-byte encoding
UTFEF	The byte is the follow-up byte in a multi-byte encoding
UTFEC	The byte is a control character (one-byte encoding)
NONUTFE	The byte is not a valid UTFE byte

CCSNAME = 7-/8-bit-code:

This specification is not permitted.

CCSNAME = UNICODE:

This specification is not permitted.

CCSNAME=UTFE:

Outputs a 256-byte properties table.



---

TOCCS	The name of the target code is specified. The name is 8 bytes long. If you do not specify a name, EDF03IRV automatically becomes the target code. This specification only makes sense for TABLE=CONVERT.
=<7-/8-bit-code>	Name of the 7-/8-bit codes.
=UNICODE	A Unicode character set is the target code.
TABADDR	Start address of the result table.
TABLEN	Address of a word which contains the length of the table supplied following successful processing of the function specified in the TABLE operand.

### 4.1.3 Functional overview

The NLSCOD (NATIONAL LANGUAGE SUPPORT: CODE TABLES) macro is activated by means of a supervisor call (SVC). NLSCOD supplies tables of varying length.

The information provided depends on the TABLE parameter:

- If both the source character set (CCSNAME) and the target code (TOCCS) are 7-/8-bit codes, the following applies:

The table supplied is 256 bytes long. The nth byte of the table supplied contains information on the character of the source character set which has the value "n".

- If TABLE=CONVERT is specified, the byte contains the target code (TOCCS) for this character. The source and target codes must be compatible.
- If TABLE=TOUPPER is specified, the byte contains the position of the corresponding uppercase letter in the source character set (CCSNAME).
- If TABLE=TOLOWER is specified, the byte contains the position of the corresponding lowercase letter in the source character set (CCSNAME).
- If TABLE=SORT is specified, the byte contains the collating sequence number of the character.
- In a list of properties (<prop>), the byte is a sequence of bits, each of which represents a property.  
Binary 1 means that the property applies; binary 2 means that it does not apply. Up to 8 properties can be specified.

#### *Example*

In EBCDIC.DF.04, the letter a has the value X'81'. This means that byte X'81' in this table contains information about the letter a of the source character set.

Source character set EBCDIC.DF.04

	7	8	9
0			
1		a	
⋮			
⋮			

Hex	Table
0	
1	
⋮	
⋮	
81	Info about a
⋮	
⋮	
FF	

- When the source character set (CCSNAME) or the target code (TOCCS) is UNICODE, the size and structure of the table supplied depend on the TABLE parameter.

For details on this, please refer to the description of the various values of the TABLE parameter.

- When the source character set (CCSNAME) is UTFE, the following values are supported for TABLE:

- The byte properties table (<utfeprop>)

The table supplied is 256 bytes long. It contains the byte properties for each byte. The following properties are possible:

- The byte is the first byte in a one-byte encoding.
- The byte is the first byte in a multi-byte code (two-, three-, four- or five-byte encoding).
- The byte is a follow-up byte in a multi-byte encoding.
- The byte is a control character (one-byte encoding).
- The byte is not a valid UTFE byte.

- TOUPPER / TOLOWER for the one-byte encodings

The table supplied is 256 bytes long.

It contains the corresponding uppercase letters for each one-byte encoding which corresponds to a lowercase letter (or vice versa). All other mappings remain the same.

#### 4.1.4 Return codes in the standard header

SUBCODE		MAINCODE		Meaning
2	1	2	1	
X' 00'	X' 00'	X' 00'	X' 00'	Processed successfully Required table available
X' 00'	X' 00'	X' 00'	X' 24'	Processed successfully; required conversion table available Note: Some characters in the source code have no equivalent in the target code.
X' 00'	X' 01'	X' 00'	X' 04'	Name of the code (CCSN) unknown
X' 00'	X' 01'	X' 00'	X' 14'	Inconsistent parameter list
X' 00'	X' 01'	X' 00'	X' 20'	Initial code (CCSNAME) and target code (TOCCS) are incompatible
X' 00'	X' 01'	X' 00'	X' 40'	Property invalid
X' 00'	X' 01'	X' 00'	X' 44'	Table invalid
X' 00'	X' 01'	X' 00'	X' 80'	Memory allocation error
X' 00'	X' 01'	X' FF'	X' FF'	The requested function is not supported; unrecoverable error
X' 00'	X' 02'	X' FF'	X' FF'	The requested function is not available Unrecoverable error (possible cause: XHCS-SYS is missing from the current configuration)
X' 00'	X' 03'	X' FF'	X' FF'	The specified version of the interface is not supported (incorrect entry for VERSION in the standard header). Unrecoverable error
X' 00'	X' 04'	X' FF'	X' FF'	The parameter list is not aligned on a word boundary.
X' 00'	X' 20'	X' 00'	X' C0'	Incorrect code tables
X' 00'	X' 41'	X' FF'	X' FF'	The subsystem is not present and must be generated.
X' 00'	X' 42'	X' FF'	X' FF'	The caller is not connected to this interface; the connection must be explicitly established.
X' 00'	X' 81'	X' FF'	X' FF'	Subsystem currently not available
X' 00'	X' 82'	X' FF'	X' FF'	Subsystem in DELETE or HOLD state

Explanation of the different fields:

SUBCODE 1	Error class indicated
= X'00':	Processed successfully
≠ X'00':	Error
SUBCODE 2	Always set to the value X'00'.
MAINCODE 1	Error information for the application program, which the latter can use to counteract operating errors.
MAINCODE 2	Currently not assigned; set to the value X'00'. If there are errors in the standard header specification used to identify the product (e.g. wrong version), the value is set to X'FF'.



When the MAINCODE is X'0024', you must use the conversion table with particular care, since some characters from the source code (e.g. letters with an accent) do not occur in the target code. If the tables are defined properly, the associated value in the conversion table is X'00' (NUL).

*Example*

```

EXMPL  START
*
STRUCT NLSCOD MF=D -----(01)
*
EXMPL  CSECT
*
        BALR  10,0
        USING *,10
        USING STRUCT,11
        LA    11,PL
*
        NLSCOD MF=E,PARAM=PL -----(02)
*
* Modification of the parameter list
        MVC   GNLTCCSN,=CL8'IS088591' -----(03)
        LA    5,LOWUP2 -----(04)
        ST    5,GNLTTADD
*
        NLSCOD MF=E,PARAM=PL -----(05)
*
        TERM
*
PL      NLSCOD MF=L,CCSNAME=EDF041, TABLE=TOUPPER, TABADDR=LOWUP1 --(06)
*
        DS    0D
LOWUP1  DS    XL256 -----(07)
LOWUP2  DS    XL256 -----(08)
*
        END

```

- (01) The structure of the parameter list is defined (DSECT).
- (02) XHCS is called with a parameter list initialized by means of MF=L.
- (03) The code name is modified.
- (04) The table address is modified.
- (05) XHCS is called with the modified parameter list.
- (06) The parameter list is declared and initialized. As a result of initialization, the table for converting lowercase letters to uppercase is requested for the EBCDIC.DF.04-1 character set table.
- (07) Table no. 1
- (08) Table no. 2



**PREFIX**

=x specifies the first character of names defined with MF=D or MF=C. The default value is PREFIX=G.

**MACID**

=xxx specifies the 2nd, 3rd, and 4th characters of names defined with MF=C. The default value is MACID=NLI (also for MF=D).

**PARAM**

=<paramlist> specifies the name of the parameter list.

=(register) contains the address of the parameter list.

**CCSNAME**

specifies the code name. The name is not more than 8 bytes in length.

If you do not specify a name, EDF03IRV is used automatically.

=<7-/8-bit-code> Name of a 7-bit or 8-bit code.

=<unicode-variant>

Name of the Unicode source code.

You can specify UNICODE, UTF16 or UTF8 as the Unicode variant or UTFE for UTF-EBCDIC. The specification of 'UNICODE' means the same as UTF16. The list output contains the names of all 7-bit or 8-bit codes defined in the system, including the three Unicode variants UTF-16, UTF-8 and UTFE.

=\*SYSDEF The system default code is used.

=\*USRDEF The user default code is used.

**ISOVAR**

=<iso-variant> specifies the number of an ISO code variant for which a list of compatible codes is supplied. The number must be an integer.

=\*ALL The names of all 7-bit and 8-bit codes defined in the system are output, including the three Unicode variants UTF-16, UTF-8 and UTFE.



## INFO

Type and scope of the output information.

Each entry relating to output information has the following structure:

Byte 0 - 7 Code name (CCS)

Byte 8 ' ' if the code is fully compatible with the corresponding ISO code variant ISO 8859.  
'P' if the code is partially compatible.

Byte 9 'I' (ISO) or 'E' (EBCDIC) depending on the type of code.

If the entry is for a Unicode variant, byte 9 is assigned the following value (see also [page 83](#)):

```
UTF16  'I'
UTF8   'I'
UTFE   'E'
```

If the INFO operand is not specified, the default value \*LIST-OF-CCS is used.

=\*LIST-OF-CCS A list of compatible codes is supplied. One entry is output per code.

This operand value supplies the same interface behavior as XHCS V1.5.

=\*ONLY-SPECIFIED-CCS

The entry for the specified CCS is output.

The INFO=\*ONLY-SPECIFIED-CCS operand is only permitted in conjunction with the CCSNAME operand.

=\*REFERENCE-CODE

When the ISOVAR operand is specified, the reference code of the ISO code variant specified is output.

ISOVAR = \*ALL is not permitted. When ISOVAR = \*ALL, return code X'00010014' is issued.

When the CCSNAME operand is specified, the reference code of the ISO code variant to which the specified CCS belongs is output.

When CCSNAME = <unicode-variant>, UTFE is output as the reference code.

LSTADDR	<p>Address of the output list. This address must begin on a halfword boundary.</p> <p>The structure of the output list depends on the specification in the LSTLEN operand:</p> <p>LSTLEN is not specified: The output list is supplied in V format:</p> <p>Byte 0 - 1 Total length of the string (including the header). The maximum length of the output list (including the header) must be entered here before the call. After the call, the two bytes contain the actual length (including the header).</p> <p>Byte 2 - 3 are filled with binary zeros</p> <p>Byte 4 - n Data</p> <p>LSTLEN is specified: The output list only contains the data.</p>
LSTLEN	<p>Address of a word which contains the length of the output list.</p> <p>The output list is supplied to the address specified using LSTADDR without a preceding length field.</p> <p>Before the call, the word whose address is specified in LSTLEN must contain the maximum length of the output list. After the call, this word contains the actual length of the output list.</p> <p>If you do not specify LSTLEN, the output list is supplied in V format.</p>

### 4.2.3 Notes on programming

The NLSCMP (NATIONAL LANGUAGE SUPPORT : COMPATIBILITY TABLE) macro is called by means of a supervisor call (SVC). NLSCMP provides a list of compatible codes. You must specify the name of a character set or the number of an ISO code variant.

If you specify a character set name (CCSNAME), the number of the corresponding ISO code variant is passed to the parameter list (&PR.ISON structure element). If you specify a Unicode variant as the character set name, i.e. UNICODE, UTF16, UTF8 or UTFE, the value 240 = X'F0' is returned as the ISO code variant. The structure element begins by default with the letters 'GNLI'. However, you can replace these letters with a string of your own.

If you specify CCSNAME = <unicode-variant> or ISOVAR=\*ALL, the output list contains not only the names of all the 7- and 8-bit codes defined in the system, but also the three Unicode variants UTF-16, UTF-8 and UTFE.

ISOVAR can stand for both ISO and EBCDIC codes.

The output list must begin on a halfword boundary. Its structure depends on whether you have specified the LSTLEN parameter:

Structure when LSTLEN is not specified (V format):

- Byte 0 - 1 Total length of the list (including the header)
- Byte 2 - 3 Reserved
- Byte 4 - n Data

Structure when LSTLEN is specified:

- Byte 0 - n Data

The data component consists of a sequence of strings 10 bytes in length, of which the first 8 bytes contain the code name (CCSN). Byte 9 is set to the value 'L' providing the code is fully compatible with the corresponding ISO code variant ISO8859. If the compatibility of the code is restricted, the 9th byte contains a 'P' (see [section “Grouping compatible codes” on page 115](#)).

The 10th byte of the 10-byte entries contains the type of code:

Type of code	Value
ISO code	'I' (ISO)
EBCDIC code	'E' (EBCDIC)
UTF-16	'I' (ISO)
UTF-8	'I' (ISO)
UTFE	'E' (EBCDIC)

## 4.2.4 Return codes

SUBCODE		MAINCODE		Meaning
2	1	2	1	
X' 00'	X' 00'	X' 00'	X' 00'	Processed successfully; required data available
X' 00'	X' 01'	X' 00'	X' 04'	Name of the code (CCSN) or ISO variant unknown
X' 00'	X' 01'	X' 00'	X' 14'	Inconsistent parameter list
X' 00'	X' 01'	X' 00'	X' 80'	Memory allocation error
X' 00'	X' 01'	X' 00'	X' 84'	The output table is too small. The data is truncated.
X' 00'	X' 01'	X' 00'	X' 88'	Length of string invalid
X' 00'	X' 01'	X' 00'	X' 8C'	Error in alignment on the halfword boundary
X' 00'	X' 01'	X' FF'	X' FF'	The requested function is not supported. Unrecoverable error
X' 00'	X' 02'	X' FF'	X' FF'	The requested function is not available. Unrecoverable error (possible cause: XHCS-SYS is missing from the current configuration)
X' 00'	X' 03'	X' FF'	X' FF'	The specified version of the interface is not supported (incorrect entry for VERSION in the standard header). Unrecoverable error
X' 00'	X' 04'	X' FF'	X' FF'	The parameter list is not aligned on a word boundary.
X' 00'	X' 20'	X' 00'	X' C0'	Invalid code tables
X' 00'	X' 41'	X' FF'	X' FF'	The subsystem is not present and must be explicitly generated.
X' 00'	X' 42'	X' FF'	X' FF'	The caller is not connected to this interface; the connection must be explicitly established.
X' 00'	X' 81'	X' FF'	X' FF'	Subsystem currently not available
X' 00'	X' 82'	X' FF'	X' FF'	Subsystem in DELETE or HOLD state

Explanation of the different fields:

SUBCODE 1	Error class indicated
= X'00':	Processed successfully
≠ X'00':	Error
SUBCODE 2	Always set to the value X'00'.
MAINCODE 1	Error information for the application program, which the latter can use to counteract operating errors.
MAINCODE 2	Currently not assigned; set to the value X'00'. If there are errors in the standard header specification used to identify the product (e.g. wrong version), the value is set to X'FF'.

*Example*

```

EXMPL  START
*
        BALR  10,0
        USING *,10
*
        NLSCMP MF=E,PARAM=PL -----(01)
*
*
        TERM
*
*
PL      NLSCMP MF=L,CCSNAME=*SYSDEF,LSTADDR=LSTCOMP -----(02)
*
*
*      Memory area of the list of compatible codes (V format)
*
        DS      0H
LSTCOMP DC      Y(LSTEND-LSTCOMP)----- (03)
RES     DC      Y(0) ----- (04)
DATA   DS      CL60 ----- (05)
LSTEND EQU     *
*
*
        END

```

- (01) XHCS call.
- (02) The parameter list is declared and initialized. A list of codes compatible with the system default is requested.
- (03) Maximum length.
- (04) Reserved.
- (05) Data component.

## 4.3 String conversion: NLSCNV

### 4.3.1 Macro call format

Operation	Operands
NLSCNV	MF=D      [,PREFIX= <prefix>] [,XPAND= <u>PARAM</u> / LOADCB]
	MF=C      [,PREFIX= <prefix>][,MACID = <macid>] [,XPAND= <u>PARAM</u> / LOADCB]
	MF=E      [,MODE= <u>*SVC</u> / *BASR / *LOAD] ,PARAM=<paramlist> / (register)
	<b>Action: Converting strings to compatible target code</b>
	MF=L / <u>S</u> [,ACTION= <u>CONVERT</u> ] ,CCSNAME= <7-/8-bit-code> / <unicode-variant> / *SYSDEF / *USRDEF ,TOCCS= <7-/8-bit-code> / <unicode-variant> / *SYSDEF / *USRDEF ,INSTRG= <addr> [,INLEN= <addr>] ,OUTSTRG= <addr> [,OUTLEN= <addr>] [,DEFAULT= <u>*STD</u> / *NO / *UTF16 / *TARGET / *SPECIAL ] [,DEFVAL= <u>*NONE</u> / xx / xxxx / xxxxxx ] [,DEFLEN= <u>*NONE</u> / 1 / 2 / 3 ] [,TABADDR= <u>*NONE</u> / <tabaddr>]
	<b>Action: Converting lowercase letters to uppercase letters and vice versa</b>
	MF=L / <u>S</u> ,ACTION= TOUPPER / TOWER ,CCSNAME= <7-/8-bit-code> / <unicode-variant> / *SYSDEF / *USRDEF ,INSTRG= <addr> [,INLEN= <addr>] ,OUTSTRG= <addr> [,OUTLEN= <addr>] [,DEFAULT= <u>*STD</u> / *NO / *UTF16 / *TARGET ] [,DEFVAL= <u>*NONE</u> / xx / xxxx / xxxxxx ] [,DEFLEN= <u>*NONE</u> / 1 / 2 / 3 ] [,TABADDR= <u>*NONE</u> / <tabaddr>]

Operation	Operands
NLSCNV	<p><b>Action: Normalizing the input string</b></p> <pre>MF=L / <u>S</u> ,ACTION= COMPOSE / DECOMPOSE            ,INSTRG= &lt;addr&gt;            [,INLEN= &lt;addr&gt;]            ,OUTSTRG= &lt;addr&gt;            [,OUTLEN= &lt;addr&gt;]            [,DEFAULT= <u>*STD</u> / *NO / *UTF16 / *TARGET ]            [,DEFVAL= <u>*NONE</u> / xx / xxxx / xxxxxx ]            [,DEFLEN= <u>*NONE</u> / 1 / 2 / 3 ]</pre> <p><b>Action: Outputting the length of UTF-8 or UTFE strings</b></p> <pre>MF=L / <u>S</u> ,ACTION= LENGTH            ,CCSNAME= &lt;7-/8-bit-code&gt; / &lt;unicode-variant&gt; /                    *SYSDEF / *USRDEF            [,TOCCS= &lt;7-/8-bit-code&gt; / &lt;unicode-variant&gt; /                    *SYSDEF / *USRDEF ]            ,INSTRG= &lt;addr&gt;            [,INLEN= &lt;addr&gt;]            ,LENGTH= &lt;addr&gt;</pre>



### 4.3.2 Operand description

#### MF

- =D generates a DSECT.
- =E generates a command component. No commands are generated for interpreting the return code.
- =C generates the layout of the NLSCNV in the current data structure. Each field is named and all equates are generated. The standard header is not set.
- =S generates a parameter list. The fields are filled in accordance with the specified keyword parameters of the macro. Names and equates are not generated. A command component is generated. The standard header is set automatically. No commands are generated for interpreting the return code.
- =L generates a parameter list. The fields are filled in accordance with the specified keyword parameters of the macro. Names and equates are not generated. The standard header is set automatically.

#### PREFIX

- =x specifies the first character of names defined with MF=D or MF=C. The default value is PREFIX=G.

#### MACID

- =xxx specifies the 2nd, 3rd and 4th characters of names defined with MF=C. The default value is MACID=NLC (also for MF=D).

#### XPAND

- specifies the data structure which is to be expanded. If you do not specify XPAND, XPAND= PARAM is used.
- XPAND is only offered when MF=D / C.
- =PARAM The NLSCNV parameter list is expanded.
- =LOADCB The control block for the branch using MODE= \*LOAD is expanded.

#### PARAM

- =<paramlist> specifies the name of the parameter list.
- =(register) contains the address of the parameter list.

MODE	<p>specifies how the component which executes the function is to be branched to (see also <a href="#">page 83</a>).</p> <p>The operand is only evaluated when MF=E.</p> <p>If you do not specify the MODE operand, the value *SVC is used.</p>
=*SVC	<p>An SVC is generated.</p> <p>R1 is loaded with the address of &lt;paramlist&gt; by the macro. When PARAM = &lt;paramlist&gt;, specification of the NLSCNV parameter list is expected.</p>
=*BASR	<p>The GNLCNV entry which executes the function is branched to directly.</p> <p>The LLM GNLCNV must be permanently linked to the user program. The LLM GNLCNV is contained in the module library SYSLNK.XHCS-SYS.020.TU.</p> <p>This mode is intended for user programs which execute exclusively in class-6 memory.</p>



This mode requires that the calling program should be ILCS-capable, and that the addressing mode ANY (= 31-bit-capable) is being used.

The following commands are generated:

```
L      R15, = V(GNLCNVC)
LA     1, <addrparamlist>
BASR  R14, R15
```

The following registers are required when the NLSCNV macro is called with MODE=\*BASR:

- R1 is loaded with the address of <addrparamlist> by the macro. When PARAM = <addrparamlist>, the address of a field containing the address of the NLSCNV parameter list is expected. Note the difference to MODE=\*SVC; there the address of the NLSCNV parameter list is specified directly in R1.
- R13 must be loaded before the macro call with the address of an 18-word-long save area which the calling program must provide.
- R14 is loaded with the return address of the user program by the macro.
- R15 = V (GNLCNVC)

=\*LOAD

The reentrant adapter module GNLADPT is branched to which dynamically loads the GNLCNV module which executes the function into class-6 memory from the module library SYS-LNK.XHCS-SYS.020.TU using BIND, and subsequently branches there.

The adapter module GNLADPT must be linked to the user program. This adapter module is available as an R module and is contained in the SYSOML.XHCS-SYS.020 library.

This mode is intended for user programs in which the NLSCNV interface is called at a point which must have reentrant capability.

When the command component is generated, a V constant is generated as with MODE=\*BASR, but GNLADPT has reentrant capability and can therefore be located at the same point for all tasks.

Dynamic loading only takes place when the first call with MODE=\*LOAD is made. i.e. when the address in the LOAD control block GNLCLCB (see below and on [page 106](#)) has not yet been assigned.

In this mode it is not essential that the calling program is ILCS-capable and that the addressing mode ANY (= 31-bit-capable) is being used.

The following commands are generated:

```
L      R15, = V(GNLADPT)
LA     1,<addrcontrolblock>
BASR  R14, R15
```

The following registers are required when the NLSCNV macro is called with MODE=\*LOAD:

- R1 is loaded with the address of <addrcontrolblock> by the macro. When PARAM = <addrcontrolblock>, specification of a 20-byte-long local task control block containing the following information is expected :
- Address of the GNLCNV entry which is assigned by GNLADPT
  - Indicator showing whether a value has been assigned to this address field, i.e. whether dynamic loading has already taken place
  - Address of the NLSCNV parameter list
  - Return code

The layout of this control block can be generated using XPAND = LOADCB.

In the case of the first call with MODE=\*LOAD, all fields of the LOAD control block (except the address of the NLSCNV parameter list) must be prefilled with 0. The adapter module GNLADPT assigns valid values to the remaining fields.

The control block with values assigned by GNLADPT must always be specified in the following calls.

The calling program should ensure that the control block assigned values by GNLADPT is retained for a further NLSCNV call with MODE=\*LOAD.

- R13 must be loaded before the macro call with the address of an 18-word-long save area which the calling program must provide.
- R14 is loaded with the return address of the user program by the macro.
- R15 = V (GNLADPT)

The return code of the NLSCNV function is supplied in the default header of the NLSCNV parameter list. The return code for loading GNLCNV dynamically using BIND is supplied in the return code field of the LOAD control block.

#### ACTION

specifies the required action.

Detailed information on this is provided in the [section “Functional overview” on page 101](#).

CCSNAME	specifies the name of the source code. The name is not more than 8 bytes in length. If you do not specify a name, EDF03IRV automatically becomes the source code.  When ACTION=LENGTH: CCSNAME must be specified. There is no default.
=<7-/8-bit-code>	Name of a 7-bit or 8-bit source code.
=*SYSDEF	The system default code is the source code.
=*USRDEF	The user default code is the source code.
=<unicode-variant>	Name of the Unicode source code. You can specify UNICODE, UTF16 or UTF8 as the Unicode variant or UTFE for UTF-EBCDIC. The specification of 'UNICODE' means the same as UTF16.  When ACTION=LENGTH: Only UTF16 or, in the case of the length calculation of predefined strings, only UTF8 or UTFE strings.
TOCCS	The name of the target code is specified. The name is not more than 8 bytes in length. If you do not specify a name, EDF03IRV automatically becomes the target code.
=<7-/8-bit-code>	Name of a 7-bit or 8-bit target code.
=*SYSDEF	The system default code is the target code.
=*USRDEF	The user default code is the target code.
=<unicode-variant>	Name of the Unicode target code. You can specify UNICODE, UTF16 or UTF8 as the Unicode variant or UTFE for UTF-EBCDIC. The specification of 'UNICODE' means the same as UTF16.  When ACTION=LENGTH: You can only specify UTF8 or UTFE. All other specifications are rejected with an error.

---

INSTRG	<p>Address of the input string.</p> <p>The structure of the input string depends on the specification in the INLEN operand:</p> <p>INLEN is not specified: The input string is expected in V format: Byte 0-1: Total length of the string (including the header) Byte 2-3: Reserved Byte 4-n: String The length of the input string may be no more than 32767 bytes. The address of the input string must begin on the halfword boundary.</p> <p>INLEN is specified: The input string contains only the characters to be processed. The length of the input string may be no more than 65536 bytes. The address of the input string need not necessarily begin on the halfword boundary.</p>
INLEN	<p>Address of a word which contains the length of the input string. This value may be no more than 65536 bytes.</p> <p>The input string is expected at the address specified by INSTRG without a preceding length field.</p> <p>If you do not specify INLEN, the input string is expected in V format.</p>

OUTSTRG	<p>Address of the output string.</p> <p>The structure of the output string depends on the specification in the OUTLEN operand:</p> <p>OUTLEN is not specified:          The output string is supplied in V format:          Byte 0 - 1: Total length of the string (including the header)          Byte 2 - 3: Filled with binary zeros          Byte 4 - n: String          The address of the output string must begin on the halfword boundary.</p> <p>OUTLEN is specified:          The output string contains only the converted characters.          The address of the output string need not necessarily begin on the halfword boundary.</p>				
OUTLEN	<p>Address of a word which, after successful conversion, contains the length of the output string.</p> <p>The output string is supplied at the address specified by OUTSTRG without a preceding length field.</p> <p>Before the call, the word whose address is specified in OUTLEN must contain one of the two values below:</p> <table style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;">0</td> <td>No specification of the maximum size of the output string</td> </tr> <tr> <td>value &gt; 0</td> <td>Specifies the maximum possible size of the output string</td> </tr> </table> <p>After the call, this word contains the actual value of the output string. The following applies here: If you have specified DEFAULT=*NO and processing is interrupted because an invalid character occurs (i.e. with return code X'00010028'), this word contains the length of the output string generated up to this point (in bytes). If the output string is truncated because the specified maximum length is reached, return code X'00000084' is issued.</p> <p>If you do not specify OUTLEN, the output string is supplied in V format.</p>	0	No specification of the maximum size of the output string	value > 0	Specifies the maximum possible size of the output string
0	No specification of the maximum size of the output string				
value > 0	Specifies the maximum possible size of the output string				

## DEFAULT

Type of wildcard character processing

XHCS V2.0 enables a wildcard character to be specified which is to be used in the output string if one of the following cases occurs:

- A character in the input string is not defined in the source CCS.
- The source CCS is a Unicode variant and a character in the input string is not a Unicode character supported by XHCS.
- In the CONVERT action, a character in the input string is not defined in the target CCS.

The following is meant by "target CCS":

<b>When ACTION=</b>	<b>Target CCS</b>
CONVERT	The CCS specified with TOCCS CCS
TOUPPER /TOWER	The CCS specified with CCSNAME
COMPOSE /DECOMPOSE	UTF-16

If you do not specify DEFAULT, the value \*STD is used.

## =\*STD

The value C'.' ( = U+002E "full stop" ) is used as a wildcard character. XHCS converts this value to the required target CCS.

Specifications for the DEFVAL and DEFLEN operands are ignored.

*Note on compatibility with XHCS V1.5*

When an undefined character occurs while converting 7-/8-bit code to 7-/8-bit code in XHCS V1.5, a NUL character is set.

If you execute a program which has been compiled using the macros from XHCS V1.5, i.e. when XHCS receives an NLSCNV parameter list in the layout of XHCS V1.5, the NUL character is used as the wildcard character for compatibility reasons in order to implement the previous behavior.

If you recompile an old program source using the macros from XHCS V2.0 without modifying the source, i.e. in particular without specifying the new DEFAULT operand, the period is used as the wildcard character. This is not compatible with the previous version.



- =\*NO** No wildcard character is used. If a character is found which does not occur in the specified code or which is not supported by XHCS, conversion is aborted and the return code X'00010028' is supplied.
- The output string contains the characters converted up to this point.
- The length of the output string (in bytes) which was generated up to the point the operation was aborted is contained
- in the length field at the beginning of the output area if the OUTLEN operand was not specified.
  - at the address specified in OUTLEN if the OUTLEN operand was specified.
- The relative position of the invalid character in the string is contained in the output field &PR.PIC1 (Position of Invalid Character in Input String). In all other cases this output field contains the value 0. Note that an offset in bytes is not supplied here but the position of the character in the input string where the invalid character is located.
- Example*
- In the UTF-8 input string C2A0C3A2CCCC the invalid character CCCC is in position 3, i.e. after the two valid UTF-8 characters C2A0 and C3A2 the 3rd character in the string is invalid.
- Specifications for the DEFVAL and DEFLEN operands are ignored.
- =\*UTF16 /\*TARGET** A freely selectable wildcard character can be used.
- =\*UTF16** The value specified in the DEFVAL operand is interpreted as a UTF-16 value regardless of the source and target CCSs selected. XHCS converts the value back to the required target CCS.
- If the specified character is not defined in the target CCS, a negative return code is supplied.
- The DEFVAL operand is expected in the form xxxx.
- Example*
- The specification DEFVAL=003F is interpreted as U+003F (= C'?' ).
- The DEFLEN operand is ignored

- =\*TARGET**            The value specified in the DEFVAL operand is interpreted as a character of the target CCS.
- The DEFVAL and DEFLLEN operands are evaluated. If the length of the wildcard character specified in DEFLLEN does not match the target CCS (e.g. target CCS = 8-bit code and DEFLLEN=2, or target CCS = UTF16 and DEFLLEN=1, etc.), a negative return code is supplied.
- Otherwise the value specified in DEFVAL is accepted unchanged with the length specified using DEFLLEN.
- Example*
- Target CCS = EDF041, DEFVAL = 6F, DEFLLEN = 1  
 Wildcard character is C'?'  
 Target CCS = UTF16, DEFVAL = 003F, DEFLLEN = 2  
 Wildcard character is C'?'
- =\*SPECIAL**            Special conversion for unambiguously reversible conversion of wildcard characters (see also the [section “Unambiguously reversible conversion for all existing 7-/8-bit character sets” on page 107](#)).
- This value may only be specified when ACTION = CONVERT and
- Source CCS = 7- / 8-bit code and target CCS = UNICODE / UTF16
  - or
  - Source CCS = UNICODE /UTF16 and target CCS = 7- / 8-bit code
- A different wildcard character from the range U+E000 through U+F8FF ("for private use") is used for each undefined code point in the 7- or 8-bit source CCS, namely the character U+F200 + <undefined code point>. This assignment enables the corresponding characters to be mapped and mapped back again.
- Example*
- If the code point X'ED' is not defined, the value X'F2ED' is used.
- When the output string generated in this way is converted back to the original source CCS, the wildcard characters are once more converted to the original value. This is an undefined code point of the source CCS.
- Specifications for the DEFVAL and DEFLLEN operands are ignored.

DEFVAL	<p>Value of the required wildcard character which is to be used in the output string.</p> <p>The operand is mandatory when DEFAULT=*UTF16 / *TARGET. The operand is ignored when DEFAULT=*STD / *NO / *SPECIAL.</p> <p>= xx specifies a hexadecimal one-byte value.</p> <p>= xxxx specifies a hexadecimal two-byte value.</p> <p>= xxxxxx specifies a hexadecimal three-byte value.</p> <p>The specified value is entered left-justified in the &amp;PR.DVAL field.</p> <p><b>Caution</b> If you do not specify the value by means of MF=L (Ass) but specify a value for the relevant field in the parameter list by means of an explicit assignment, specify the value &amp;PR.DU16 or &amp;PR.DTRG in the &amp;PR.DIND field. If you have specified &amp;PR.DTRG, the length field &amp;PR.DLEN must also be supplied with a value.</p>
DEFLEN	<p>specifies the length of the wildcard character.</p> <p>The operand is mandatory when DEFAULT=*TARGET. It is ignored for all other DEFAULT specifications.</p> <p>When DEFLEN=n, the first n bytes in the &amp;PR.DVAL field are interpreted as wildcard characters.</p>
LENGTH	<p>When ACTION=LENGTH: Address of the word into which the length determined is to be written.</p>

TABADDR	<p>Address of a conversion table.</p> <p>The TABADDR operand enables you obtain a conversion table for a particular action and then to specify the appropriate conversion table for each action using TABADDR, which means that XHCS idoes not need to use the SVC to obtain the conversion table when you switch between actions. However, to permit this you must also manage the various conversion tables yourself.</p> <p>If you do not specify the TABADDR operand, the value *NONE is used.</p> <p>If the input string is normalized (ACTION = COMPOSE / DECOMPOSE), the TABADDR operand is ignored.</p>												
= *NONE	<p>No conversion table is made available for XHCS, nor is a conversion table provided by XHCS. XHCS obtains the required conversion table itself.</p>												
= <tabaddr>	<p>Address of an area consisting of two words</p> <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 10px;">1st word</td> <td style="padding-right: 20px;">Meaning of the address in the 2nd word</td> <td></td> </tr> <tr> <td style="padding-right: 10px;">-1</td> <td style="padding-right: 20px;"></td> <td>The 2nd word contains the address of a table which is to be used by XHCS as the conversion table for the required NLSCNV action.</td> </tr> <tr> <td style="padding-right: 10px;">Value ≠ -1</td> <td style="padding-right: 20px;"></td> <td>The conversion table used by XHCS in the required NLSCNV action is to be copied to the address specified in the 2nd word.</td> </tr> <tr> <td style="padding-right: 10px;">2nd word</td> <td style="padding-right: 20px;"></td> <td>Address at which the conversion table is expected or to which the conversion table is to be copied.</td> </tr> </table>	1st word	Meaning of the address in the 2nd word		-1		The 2nd word contains the address of a table which is to be used by XHCS as the conversion table for the required NLSCNV action.	Value ≠ -1		The conversion table used by XHCS in the required NLSCNV action is to be copied to the address specified in the 2nd word.	2nd word		Address at which the conversion table is expected or to which the conversion table is to be copied.
1st word	Meaning of the address in the 2nd word												
-1		The 2nd word contains the address of a table which is to be used by XHCS as the conversion table for the required NLSCNV action.											
Value ≠ -1		The conversion table used by XHCS in the required NLSCNV action is to be copied to the address specified in the 2nd word.											
2nd word		Address at which the conversion table is expected or to which the conversion table is to be copied.											

### 4.3.3 Functional overview

The functions of the NLSCNV macro are dependent on the ACTION operand:

<b>ACTION=</b>	<b>Meaning</b>	<b>See also</b>
CONVERT	Converts strings to compatible target code	<a href="#">page 101</a>
TOUPPER/TOLOWER	Converts lowercase letters to uppercase letters and vice versa	<a href="#">page 103</a>
COMPOSE/DECOMPOSE	Normalizes input string	<a href="#">page 103</a>
LENGTH	Outputs length of UTF-8 or UTFE strings	<a href="#">page 104</a>

#### **ACTION=CONVERT: Converting strings to compatible target code**

NLSCNV converts strings directly from their source code (CCSNAME) to the compatible target code (TOCCS).

*Neither the source code nor the target code is Unicode:*

The input and output strings must begin on the halfword boundary and have the following structure (V format):

The input and output strings can occupy the same memory area either partially or fully (INSTRG = OUTSTRG).

The length of the output string is equal to the length of the input string.

*For strings which are encoded using Unicode:*

In addition to the conversions between compatible 8-bit codes, the following additional conversion options are also available:

- A specified 7-bit or 8-bit code (CCSNAME) can be converted to a Unicode variant (TOCCS).

Ensure that the output area is large enough since the output length can be up to three times that of the input length, see also the tables below.

- A Unicode string can be converted into a specified 7-bit or 8-bit code. When 'UNICODE' or 'UTF16' is specified, the output length is halved, with 'UTF8' or 'UTFE' it depends on the type of characters used.

- One Unicode variant can be converted to another, e.g. a UTF-16 string to UTF-8 or UTFE and vice versa.

If a character is found which is not included in the specified 7-bit or 8-bit code or not in the Unicode support of XHCS, then

- if DEFAULT=\*NO was specified, processing is aborted and the return code X'00010028' is issued,
- if a value other than \*NO was specified for DEFAULT, the specified wildcard character (possibly converted to the target CCS) is inserted in place of the undefined character and the return code X'00010024' is issued.
- if DEFAULT was not specified, the default wildcard character is inserted and the return code X'00010024' is issued. The default wildcard character is '.'.

Maximum length of the output string ( $L_E$  = length of the input string):

Target code Source code	8-bit code	UTF-16	UTF-8	UTFE
8-bit code	$L_E$	$2*L_E$	$3*L_E$	$3*L_E$
UTF-16	$0,5 *L_E$	$L_E$	$1,5*L_E$	$2*L_E$
UTF-8	$L_E$	$2*L_E$	$L_E$	$1,5*L_E$
UTFE	$L_E$	$2*L_E$	$2*L_E$	$L_E$

When the maximum length of the input string is 64 K, the maximum possible length of the output string is as follows:

Target code Source code	8-bit code	UTF-16	UTF-8	UTFE
8-bit code	64 K	128 K	192 K	192 K
UTF-16	32 K	64 K	96 K	128 K
UTF-8	64 K	128 K	64 K	96 K
UTFE	64 K	128 K	128 K	64 K

**ACTION=TOUPPER / TOWER: Converting lowercase letters to uppercase letters and vice versa**

In strings which are encoded in accordance with the source code (CCSNAME), this function converts the lowercase letters to uppercase letters (TOUPPER) or the uppercase letters to lowercase letters (TOWER).

If a character is not included in the code set supported, the wildcard processing defined with the DEFAULT, DEFVAL and DEFLN operands comes into force:

- If DEFAULT=\*NO was specified, processing is aborted and the return code X'00010028' is issued.
- If a value other than \*NO was specified for DEFAULT, the specified wildcard character is inserted and the return code X'00010024' is issued.
- If DEFAULT was not specified, the default wildcard character '.' (=U+002E) is inserted and the return code X'00010024' is issued.

Otherwise it remains unchanged by the conversion.

The input and output strings are located at the addresses specified by INSTRG and OUTSTRG and can overlap either partially or fully.

The output string and input string are of the same length.

**ACTION=COMPOSE / DECOMPOSE: Normalizing an input string**

When this specification is entered, XHCS normalizes the input string.

- COMPOSE: Conversion to Normalization Form C
- DECOMPOSE: Conversion to Normalization Form D  
(see the [section "Unicode support" on page 18](#))



The input string is expected in UTF-16.

If a character is not included in the code set supported, the wildcard processing defined with the DEFAULT, DEFVAL and DEFLN operands comes into force:

- If DEFAULT=\*NO was specified, processing is aborted and the return code x'00010028' is issued.
- If a value other than \*NO was specified for DEFAULT, the specified wildcard character is inserted and the return code x'00010024' is issued.
- If DEFAULT was not specified, the default wildcard character '.' (=U+002E) is inserted and the return code X'00010024' is issued.

*Maximum possible length of the output string*

<b>ACTION</b>	<b>Maximum possible length of the output string</b>
COMPOSE	Length of the input string
DECOMPOSE	3 * length of the input string As the input is expected in UTF-16, a character is 2 bytes long. In the "worst" case, each input character is a character with three collation elements.

### **ACTION=LENGTH: Outputting the length of UTF-8 and UTFE strings**

This function offers you the following two options:

- The length of the relevant UTF-8 (UTFE) string is output for 8-bit and UTF-16 strings.  
For this purpose, specify the INSTRG, LENGTH, CCSNAME and TOCCS parameters. Any other specifications, such as DEFAULT or OUTSTRG, are ignored.
- Various length calculations of predefined UTF-8 (UTFE) strings

The corresponding number of characters is determined from the number of bytes in the output string, or the corresponding number of bytes in the output string from the number of characters specified.

Specify the INSTRG, LENGTH and CCSNAME parameters. Any other specifications, such as DEFAULT or OUTSTRG, are ignored.

A word is expected at the address specified by LENGTH, the first halfword (HW) containing the number of bytes and the second halfword the corresponding number of characters and vice versa. The following applies:

- 1st HW  $\neq$  0 and 2nd HW = 0: The 2nd HW contains the number of characters corresponding to the number of bytes specified in the 1st HW. The 1st HW contains the number of bytes corresponding to the number of characters specified in the 2nd HW.
- 1st HW = 0 and 2nd HW  $\neq$  0: The number of bytes specified in the 2nd HW is stored in the 1st HW.
- 1st HW  $\neq$  0 and 2nd HW  $\neq$  0: Error !
- 1st HW = 0 and 2nd HW = 0: The 1st HW contains the number of bytes in the output string. The 2nd HW contains the number of characters.



- The length specification for the output string is the highest processing length in bytes.  
If you specify a number of characters in the 2nd HW which corresponds to a number of bytes which goes beyond this processing length, processing only takes place up to this length.

### *Example*

Let us take the string `Biölä`.

Hexadecimal representation in ISO8859-1	DF69F66CC4
UTF-16 representation	00DF006900F6006C00C4
Number of characters	5

Conversion to UTF-8 yields the following: `C39F 69 C3B6 6C C384`. In other words 8 bytes.

Output string in UTF-8 with length field: `0x00080000C39F69C3B66CC384`.

If you want to determine the number of characters which correspond to 5 bytes, enter `0x00050000` in the word to which `LENGTH` points. After you have called `NLSCNV`, you receive `0x00050003`, because 5 bytes correspond precisely to 3 characters in the output string. If you enter a byte number of 4, you receive `0x00040003`, because the incomplete character is also counted.

If, on the other hand, you want to know the number of bytes which correspond to 3 characters, enter `0x00000003` in the word to which `LENGTH` points. After you have called `NLSCNV`, you receive the value `0x00050003` in this field. If, for example, you specify 7 as the number of characters, 8 is returned as the corresponding number of bytes because this is the maximum length specified in the length field of the output string.

## 4.3.4 Notes on programming

### **Branching from TU without SVC**

The `NLSCNV` (NATIONAL LANGUAGE SUPPORT: CONVERSION FUNCTION) macro can - as previously - be called by means of a supervisor call (SVC).

If you frequently call the `NLSCNV` conversion interface and complete SVC processing takes place each time, performance problems can occur

To prevent this, XHCS V2.0 and higher enables the required function to be called via a call interface, i.e. by means of `BASR`.

Even if the required functions is not branched to by means of SVC but via BASR, the entire procedure is not necessarily free of SVCs.

The number of SVCs is reduced as follows:

- When the same action is requested twice or more in succession, e.g. conversion from EDF041 to ISO88591, XHCS notes the table(s) required for this and does not fetch them again.

The SVCs are issued only when a particular action is called for the first time.

- If you use the TABADDR operand (see [page 100](#)) to specify the "actual" conversion table, it no longer needs to be fetched by XHCS.

The LLM GNLCNV is provided to branch to the required function by means of BASR. Users can link this to their user program. The LLM GNLCNV is contained in the module library SYSLNK.XHCS-SYS.020.TU.

As an alternative to permanently linking GNLCNV, XHCS enables a branch to an adapter module with reentrant capability which loads the LLM GNLCNV dynamically using BIND.

This adapter module is available as an R module and can be linked to the calling program. It is therefore located at the same address for all users of the shared code. The adapter module GNLADPT is contained in the SYSOML.XHCS-SYS.020 library.

Specifically, the adapter module GNLADPT executes the following actions:

- Save the caller's registers in the memory area provided by the caller
- Request memory for "automatic data", e.g. the BIND parameter list, using REQM
- Determine the installation path
- Load the LLM GNLCNV dynamically in class-6 memory using BIND
- Branch to entry GNLCNVC
- Set default return code when unsuccessful

The caller must provide a task-specific "control block" in which GNLADPT can place the address supplied by BIND. In addition, whether the BIND has already taken place is recorded there so that dynamic loading only takes place when the first call occurs (and not in response to all the subsequent calls). Furthermore, the control block must also contain the address of the actual NLSCNV parameter list.

The MODE and XPAND operands which have been introduced for this purpose are described on [page 89](#).

### **Unambiguously reversible conversion for all existing 7-/8-bit character sets**

There are gaps in the definition of some of the ISO codes supported and the corresponding EBCDIC codes, e.g. of ISO88597 and EDF047.

When a string which is to be converted contains characters which are not defined in the source CCS, wildcard characters can be contained in the target string. As all undefined code positions are mapped to one and the same wildcard character when conversion takes place, it is impossible to tell which character a wildcard character was generated from when converting the characters back, and as a result the source string can no longer be reconstructed.

To permit the original input characters to be inserted again when characters are reconverted, the characters in the input string which are not defined in the source CCS must be converted into *different* wildcard characters.

When you do not want to use one and the same wildcard character for conversion purposes but require conversion to be unambiguously reversible, specify the value \*SPECIAL for the DEFAULT operand.

If at least one character which is not defined in the source CCS and which needs to be converted in accordance with "special mapping" is contained in the input string, the return code X'00010024' is issued.

Details on this are provided in the description of the DEFAULT operand starting on [page 96](#) and in the [section "Return codes" on page 109](#).

## Length of the input and output strings

- The input and output strings can occupy the same memory area either partially or fully (INSTRG = OUTSTRG) if the length of the output string is the same as the length of the input string, i.e. if
  - ACTION = CONVERT and neither the source code (CCSNAME) nor the target code (TOCCS) is a Unicode variant,
  - ACTION = TOUPPER / TOLOWER regardless of whether or not the specified CCS is a Unicode variant.

In all other cases you must assume that the length of the output string is not the same as the length of the input string. In this case the input and output strings may not overlap.

- The actual length of the input string must be defined by the caller.

In addition to specifying the input and output strings in V format, you can also handle the strings using the INLEN and OUTLEN operands. These contain the address of a word which contains the length of the string.

The maximum length is

- 65536 (= X'10000') bytes if the length of the input string is specified using INLEN.
  - 32767 (= X'7FFF') bytes if the input string is specified in V format.
- The new OUTLEN operand must be used to specify a maximum value for the length of the output string.  
When V format is used, everything remains the same for the sake of compatibility.
  - After conversion has been successfully completed, the total length of the output string is entered by XHCS.

The length of the output string depends on the action specified and the required source/target code.

If the caller has specified a maximum output length using the OUTLEN operand and the output string is longer than this predefined length, the output data is truncated and the return code X'00 00 0084' ("truncated") is supplied.



### CAUTION!

If the caller has not specified a maximum output length, he/she must ensure that the memory space is long enough for the output string to guarantee that the complete, converted string can be accommodated. Otherwise user data will be lost.

Information on the maximum possible length of the output string in accordance with the function is provided in the [section "Functional overview" on page 101](#).

### 4.3.5 Return codes

SUBCODE 2   1		MAINCODE 2   1		Meaning
X' 00'	X' 00'	X' 00'	X' 00'	Processed successfully; conversion carried out
X' 00'	X' 00'	X' 00'	X' 24'	Conversion carried out successfully. Note: Some characters of the source code have no equivalent in the target code.
X' 00'	X' 00'	X' 00'	X' 84'	Output truncated. The length of the output string exceeds the maximum output length specified using the OUTLEN operand, and output data has been truncated.
X' 00'	X' 01'	X' 00'	X' 04'	Name of the code (CCSN) unknown
X' 00'	X' 01'	X' 00'	X' 20'	Initial code (CCSNAME) and target code (TOCCS) are incompatible
X' 00'	X' 01'	X' 00'	X' 24'	Conversion carried out successfully. Some characters of the input string are not defined in the source code or have no equivalent in the target code. or The source CCS is a Unicode variant and some characters in the input string are not included in the Unicode characters supported by XHCS. In the output string these characters have been replaced by the specified wildcard character.
X' 00'	X' 01'	X' 00'	X' 28'	Conversion aborted. Some characters of the input string are not defined in the source code or have no equivalent in the target code. or The source CCS is a Unicode variant and some characters in the input string are not included in the Unicode characters supported by XHCS. Processing was aborted because DEFAULT=*NO was specified.
X' 00'	X' 01'	X' 00'	X' 80'	Memory allocation error

SUBCODE 2   1		MAINCODE 2   1		Meaning
X' 00'	X' 01'	X' 00'	X' 88'	Length of the string invalid. <ul style="list-style-type: none"> <li>– The length specified in INLEN is &lt; 0 or &gt; 65536.</li> <li>– The length specified in OUTLEN is &lt; 0.</li> <li>– The length specified in the first 2 bytes in V format is &lt; 4 or &gt; 32767.</li> <li>– The input and output strings overlap and/or the target code is a Unicode variant and ACTION ≠ TOUPPER or TOLOWER.</li> </ul>
X' 00'	X' 01'	X' 00'	X' 8C'	Error in alignment on the halfword boundary
X' 00'	X' 01'	X' 00'	X' C4'	Invalid specifications in the wildcard character specification. <ul style="list-style-type: none"> <li>– The specified wildcard character is not defined in the target CCS (when DEFAULT=*UTF16).</li> <li>– The length specified in the DEFLEN operand does not match the target CCS (when DEFAULT= *TARGET).</li> <li>– Invalid value for DEFAULT</li> <li>– DEFAULT=*SPECIAL and the action is not conversion of a 7-/8-bit code to UTF-16 or vice versa.</li> </ul>
X' 00'	X' 01'	X' 00'	X' C8'	Error in ACTION=LENGTH <ul style="list-style-type: none"> <li>– Neither the source CCS nor the target CCS is UTF-8 or UTFE.</li> <li>– Both of the specification in the LENGTH operand are not equal to 0.</li> </ul>
X' 00'	X' 01'	X' FF'	X' FF'	The requested function is not supported. Unrecoverable error.
X' 00'	X' 02'	X' FF'	X' FF'	The requested function is not available. Unrecoverable error. Possible cause: XHCS-SYS is missing from the current configuration
X' 00'	X' 03'	X' FF'	X' FF'	The specified version of the interface is not supported (incorrect entry for version in the standard header). Unrecoverable error.
X' 00'	X' 04'	X' FF'	X' FF'	The parameter list is not aligned on a word boundary.
X' xx'	X' 20'	X' 00'	X' C0'	Invalid code tables
X' 00'	X' 41'	X' FF'	X' FF'	The subsystem is not present and must be explicitly generated.

SUBCODE 2   1		MAINCODE 2   1		Meaning
X' 00'	X' 42'	X' FF'	X' FF'	The caller is not connected to this interface; the connection must be explicitly established.
X' 00'	X' 81'	X' FF'	X' FF'	Subsystem currently not available.
X' 00'	X' 82'	X' FF'	X' FF'	Subsystem in DELETE or HOLD state.

Explanation of the different fields:

SUBCODE 1	Error class indicated
= X'00':	Processed successfully
≠ X'00':	Error, except with MAINCODE X'00' X'24'.
SUBCODE 2	Always set to the value X'00', except with MAINCODE X'00' X'C0'.
MAINCODE 1	Error information for the application program, which the latter can use to counteract operating errors.
MAINCODE 2	Currently not assigned; set to the value X'00'. If there are errors in the standard header specification used to identify the product (e.g. incorrect version), the value is set to X'FF'.

*Example*

```

EXMPL  START
*
        BALR  10,0
        USING *,10
*
*
        NLSCNV MF=E,PARAM=PL -----(01)
*
*
        TERM
*
*
PL      NLSCNV MF=L,CCSNAME=ISO88591,TOCCS=EDF041,INSTRG=INS,  C
        OUTSTRG=INS -----(02)
*
*
        Memory area for input and output data
*
        DS    0H
INS     DC    Y(INSTEND-INS)-----(03)
RES     DC    Y(0) -----(04)
TXT     DC    C'text to be converted'
INSTEND EQU   *
*
*
        END

```

- (01) XHCS is called with a parameter list initialized by means of MF=L.
- (02) The parameter list is declared and initialized. The western European ISO code (ISO8859-1) is converted to a western European EBCDIC code (EBCDIC.DF.04-1). The string (V format) begins at the address 'INS'.
- (03) Length of the V format.
- (04) Reserved.



---

## 5 Definition of code tables

Descriptions of the codes supported are the essential data with which XHCS works. These are defined and managed by the systems support staff.

These descriptions are in the form of tables, known as code tables, in the GNLMTAB module, which is created when an ASSEMBLER source is translated. The source of GNLMTAB is contained in the library SYSSRC.XHCS-SYS.020.GNLMTAB.

### 5.1 Code names

A code is identified by its code name (CCSN), which must not be more than 8 bytes in length. Its syntax corresponds to the SDF data type <name 1..8>. The characters A-Z, 0-9, @, # and \$ are permitted. The first byte must be an alphabetic character (A-Z, @, #, \$). If you use standard codes, please use the names in [chapter “Tables” on page 143](#). They are grouped according to their compatibility. Standard codes should not be redefined.

#### 5.1.1 System default code

The system default code is the character set provided by the system.

The default character set is an extended character set. You can choose whether you want to work with the user default code or the system default code. The system default code is defined as a class 2 system parameter called HOSTCODE. The default value for HOSTCODE is EDF03IRV.

## 5.1.2 User default character set

Systems support staff can assign each user a user default character set. The user default character set is an extended character set.

The systems support staff can use the ADD-USER or MODIFY-USER command to assign each user a user default character set. The SHOW-USER-ATTRIBUTES command is used to return the user default character set.

If you use the \*USRDEF parameter to call XHCS, the user default character set is used. If no user default character set is assigned, the system default is used instead.

We recommend using EDF03IRV as the system default code. This prevents warnings from being issued to users of 7-bit data display terminals. Users of 8-bit data display terminals should specify an 8-bit code as the user default. Users of 7-bit data display terminals should not specify any user default.

VTSU supports only EBCDIC defaults, fully compatible codes, and the EDF03IRV code (default 7-bit mode) as user default codes.

## 5.2 Grouping compatible codes

Only compatible codes can be converted from one to another, which is why code compatibility information must be stored together with the associated data.

Individual codes are combined into groups of compatible codes that are identified via (pseudo) ISO code variant number(s). These numbers originally referred to the numbers of the corresponding ISO 8859 variant; this no longer applies since the code tables for Arabic, Persian, and North African (French/Arabic) follow a different numbering scheme.

The reference code has a special status in each code group. All codes of the same code group are defined by the reference code. The reference code contains all characters that occur in the other codes of the same group; for this reason, it is also called a group reference code. You must select an EBCDIC.DF.04-n code as a reference code. When the subsystem is initialized, it checks whether this code has been specified as the reference code. The subsystem is loaded only if this is the case.

The conversion table of each individual code of the group in the associated reference code is stored so it is available (see [page 116](#)). Conversion tables have to be created to convert a code to another code of the same group or to convert a group reference code into another code. For this, XHCS uses existing tables.

A code group can contain codes that are fully compatible with an ISO 8859 code (e.g. EBCDIC.DF.04-n) or partly compatible (e.g. EBCDIC.DF.03-DRV or ISO646-IRV). Partly compatible codes consist of a subset of the characters of the corresponding ISO 8859 code. Since partly compatible codes can always be converted to the reference code whereas the reference code cannot always be converted to the partly compatible codes for all characters of a group reference code, partly compatible codes are always identified.

### *Example*

Each character defined under EBCDIC.DF.03 has an equivalent in ISO 8859. The accented letters in ISO 8859-1 do not have a counterpart in EBCDIC.DF.03-IRV itself but instead in its code extension EBCDIC.DF.04-1.



Some products use the identification of partly compatible codes to determine whether the data can be displayed on a 7-bit data display terminal. As a result, we recommend that 8-bit codes not be defined as partly compatible codes.

## 5.3 Table structure

Each 8-bit code used in a system is described by six tables, the first five of which occupy 256 bytes and the sixth table 512 bytes. The tables are grouped together in a data structure, in which the tables of the system default code are defined first. In order to support 8-bit terminals, the default code is an extended character set.

The following tables exist:

- a table for converting a code to the reference code of the family
- a table for converting lowercase letters to uppercase
- a sort table
- a properties table (which stores 8 different properties)
- a table for converting the reference code
- a table for converting 8-bit code to Unicode

The tables must comply with the rules described below. XHCS does not check whether they actually do.

### 5.3.1 Table for converting to the reference code

This table allows a code to be converted to the reference code of its family. For each character in the code to be converted, the table contains the code of the corresponding character in the family reference code. If the code contains an undefined character, this must be mapped to NUL (X'00') in the conversion table. For the reference code, the mapped position must correspond to the character value (i.e. byte no. *i* must have the character value *i*).

In accordance with the conventions, BS2000 uses a number of base symbols that have the same character value in all defined EBCDIC variants and thus cannot be redefined, i.e. the EBCDIC kernel and the following characters:

#	hash mark
@	commercial at
\$	dollar sign
!	exclamation mark
"	double quote

XHCS does not check whether the redefined character is a base symbol.



In EBCDIC character sets, character values lower than X'40' are reserved for control characters and should therefore remain unchanged.

### 5.3.2 Table for converting lowercase letters to uppercase

This table maps each lowercase letter to the position of the corresponding uppercase letter. The other characters are mapped to themselves.

### 5.3.3 Sort table

This table describes the collating sequence of the individual characters. The conversion must be unambiguous (i.e. two different characters must not be converted to the same character value), otherwise conversion in the opposite direction is not possible.

### 5.3.4 Table of character properties

This table stores eight properties for each character. Each property is coded as a bit. The property applies if the bit is set to '1'. If the bit is set to '0', it does not apply.

Each byte is represented in the table as follows, bit 0 being the least significant bit:

- Bit 0 :           The character is a letter.
- Bit 1 :           The character is a digit.
- Bit 2 :           The character is an arithmetic character.
- Bit 3 :           The character is a special character.
- Bit 4 :           The character is a lowercase letter.
- Bit 5 :           The character belongs to the EBCDIC kernel.
- Bit 6 :           The character is defined in the code.
- Bit 7 :           The character is displayable/printable.

### 5.3.5 Table for converting reference codes

This table enables the conversion of the reference code to the appropriate code as described in the six tables. For every character of the group reference code the table contains the code of the corresponding character in the target code.

In each macro delivered by default that defines an 8-bit code, the table for converting the reference code is empty, i.e. it contains 256 characters with the value X'00'. Systems support can redefine this table by modifying and recompiling GNLMTAB.

As long as the fifth table is empty, it is generated automatically by XHCS when loading the subsystem. This guarantees consistency with the table for conversion to the reference code of the group (first table).

If systems support modifies the table for converting the reference code, XHCS-SYS uses the modified table without checking for consistency with the first table. Normally the first and fifth tables should contain reciprocal values.

#### *Example*

EDF041 is the reference code of the code group containing the codes ISO88591, EDF041 and EDF04DRV.

For ISO88591 the first table defines the conversion into reference code EDF041. Address X'31' (ASCII coding of the character '1') contains the value X'F1' (EBCDIC coding of the character '1').

The fifth table defines the conversion of EDF041 into ISO88591. Address X'F8' (EBCDIC coding of the character '8') contains the value X'38' (ASCII coding of the character '8').

### 5.3.6 Table for Unicode mapping

This table assigns the corresponding Unicode position to each defined point in an 8-bit code table. It is currently defined for ISO8859-1/-2/-3/-4/-5/-7/-9/-15 and for the pseudo ISO code ISOEXT1, but can be extended for further ISO 8-bit codes. For compatible 8-bit codes, in other words in particular for EDF04-1/-2/-3/-4/-5/-7/-9/-15, the table is calculated from the corresponding ISO 8-bit tables as required.

The table is 2\*256 bytes in length: The corresponding Unicode character with a size of 2 bytes is assigned to each defined value of the 8-bit code.

## 5.4 Creation and modification of code tables

All XHCS tables are stored in the GNLMTAB module. This module is generated from an ASSEMBLER source by means of the NLSCTAB macro and must be incorporated into the SYSLNK.XHCS-SYS.020 module library or into a user specific library (see [section “Installation of user-defined tables” on page 128](#)). The NLSCTAB macro generates all the tables that support each individual code, for up to 99 codes. The GNLMTAB source is in the SYSSRC.XHCS-SYS.020.GNLMTAB library. Systems support staff can also generate their own code tables using the NLSCTAB macro by modifying standard code tables. Modifying standard code tables is similar to creating user-specific FHS code tables (MGCTS macro). The modified GNLMTAB module can be linked into the system using the /ADD-CODE-TABLES command (see the [section “Adding new coded character sets dynamically: ADD-CODE-TABLES” on page 132](#)). However, the modified GNLMTAB module cannot be used until the subsystem is loaded again, i.e. after another system start



Please note that if you modify tables, errors will occur during data transfer with other BS2000 computers unless the latter use the same tables. You must also observe the rules and conventions described on [page 133](#); for example that you should not change characters that are significant for the system (e.g. the characters of the command syntax). To avoid errors of this kind, you should use the extended character sets shipped as standard, and regard table modification as an additional XHCS option to be used only in special cases (e.g. for reasons of compatibility).

Because of Unicode support, not only the familiar tables need to be defined in such an NLSCTAB macro, but also a table for mapping 8-bit code to Unicode. This table is 2 \* 256 bytes in size and is located behind the inverse conversion table. In it each character of the 8-bit code is assigned the corresponding Unicode position.

The NLSCTAB macro should only be assembled with the H-ASSEMBLER: its parameter list can be too large for the F-ASSEMBLER (the maximum parameter length is 127 characters). The NLSHEAD and NLSCCS macros are offered as alternatives for the F-ASSEMBLER. For H-Assembler the parameter list is limited to 1020 characters. When very large table modules are generated, this may not be sufficient.

## 5.4.1 Generating sets of tables

This section explains how to generate sets of tables.

### NLSCTAB macro

Call:

```
NLSCTAB ((CCSN, isovar, refcode, fullcode, type, deftab, macroname), ...)
```

### Parameter description

CCSN	Name of the code to be defined.
isovar	ISO code variant number of the code.
refcode	Indication of the reference code *REF For the reference code of the family *NORMAL For the other codes
fullcode	Indication of the compatibility (fully/partly compatible) of individual codes (see <a href="#">section "Grouping compatible codes" on page 115</a> ). *FULL The code is fully compatible with the reference code. *RESTR The code is partly compatible with the reference code (i.e. it is a restricted code).
type	Indication of the code type *EBCDIC The code is an EBCDIC code. *ISO The code is an ISO code.
deftab	Name of the initial set of tables that the user can modify as required. In order to be independent of the NLSCTAB macro, these tables are available as individual macros. They are found in the SYSLIB.XHCS-SYS.020 library together with the NLSCTAB macro.
macroname	Name of the redefinition macro.



## 5.4.2 NLSHEAD and NLSCCS macros

The NLSHEAD and NLSCCS macros allow table definitions for up to 99 codes, both with the F-ASSEMBLER and the H-ASSEMBLER. They work similarly to the NLSCTAB macro.

### **NLSHEAD macro**

The NLSHEAD macro creates the header of the GNLMTAB module. Its only parameter is the number of codes to be declared.

Call:

```
NLSHEAD    NCCS=number-of-codes
```

*Example*

```
NLSHEAD    NCCS = 3
```

**NLSCCS macro**

The NLSCCS macro must be called once for each definition of a code. It takes as an argument a parameter list that is the same as a sublist of the NLSCTAB macro.

**Redefinition macro** (to be defined by the systems support staff)

```

                MACRO
&MACNAM macroname -----(1)
                GBLC   &GVAR1
*
&MACNAM EQU    *
*
                ORG    DEFTB&GVAR1.n+X'dist' -----(2)
                DC     X'hexcode' -----(3)
                .
                .
                .
                MEND

```

(1) Identical with the macro name in NLSCTAB

(2) DEFTB&GVAR1.n

Addresses in the base tables

n=0 Conversion table

n=1 Table for converting lowercase letters to uppercase

n=2 Sort table

n=3 Properties table

dist Distance from the beginning of the table

(3) New value of the specified byte(s)



The macro variable GVAR1 is used by NLSCTAB (or NLSCCS) to index the code to be redefined.

### 5.4.3 Example of the generation of a set of tables

This example illustrates how tables are generated for ISO8859-1, EBCDIC.DF.04-1 and EBCDIC.DF.03-DRV. EBCDIC.DF.04-1 is the reference code and system default for the extended computer code. In BS2000 V10, this default must be declared first in the data structure.

The existence of predefined macros is assumed. These macros describe EBCDIC.DF.04-1, EBCDIC.DF.03-DRV and ISO8859-1.



The redefinition macro below does not have to be generated by the systems support staff since EBCDIC.DF.03-DRV is one of the standard codes shipped with XHCS as a macro.

#### Redefinition macro

```
MACRO
&MACNAM  ITOD
          GBLC  &GVAR1
*
* Tables are created for EBCDIC.DF.03-DRV by modifying
* EBCDIC.DF.03-DRV tables
*
&MACNAM  EQU  *
*
* Redefinition of the conversion table:
* (Reference code is EBCDIC.DF.04-1)
      ORG  DEFTB&GVAR1.0+X'4F'    * ö
      DC   X'CC'
      ORG  DEFTB&GVAR1.0+X'7C'    * Paragraph character
      DC   X'B5'
      ORG  DEFTB&GVAR1.0+X'BB'    * Ä Ö Ü
      DC   X'63ECFC'
      ORG  DEFTB&GVAR1.0+X'FB'    * ä
      DC   X'43'
      ORG  DEFTB&GVAR1.0+X'FD'    * ü
      DC   X'DC'
      ORG  DEFTB&GVAR1.0+X'FF'    * ß
      DC   X'59'
*
* Redefinition of the table for converting lowercase letters to uppercase
*
      ORG  DEFTB&GVAR1.1+X'4F'
      DC   X'BC'
      ORG  DEFTB&GVAR1.1+X'FB'
```

```

DC      X'BB'
ORG     DEFTB&GVAR1.1+X'FD'
DC      X'BD'

```

\*

\*

\* Redefinition of the sort table

\* The sort table is reorganized so that umlauts follow the appropriate vowels without umlaut characters and 'ß' follows 'S'.

\* The sort order of some characters is transposed to ensure that the table remains unique.

\*

\*

```

ORG     DEFTB&GVAR1.2+X'43'
DC      X'76'
ORG     DEFTB&GVAR1.2+X'4F'
DC      X'D5'
ORG     DEFTB&GVAR1.2+X'59'
DC      X'FF'
ORG     DEFTB&GVAR1.2+X'5F'
DC      X'42'
ORG     DEFTB&GVAR1.2+X'63'
DC      X'77'
ORG     DEFTB&GVAR1.2+X'7C'
DC      X'5F'
ORG     DEFTB&GVAR1.2+X'A1'
DC      X'52'
ORG     DEFTB&GVAR1.2+X'B5'
DC      X'63'
ORG     DEFTB&GVAR1.2+X'BB'
DC      X'97D6F1'
ORG     DEFTB&GVAR1.2+X'CO'
DC      X'5D'
ORG     DEFTB&GVAR1.2+X'CC'
DC      X'5B'
ORG     DEFTB&GVAR1.2+X'DO'
DC      X'5E'
ORG     DEFTB&GVAR1.2+X'DC'
DC      X'5CEC'
ORG     DEFTB&GVAR1.2+X'EO'
DC      X'6A'
ORG     DEFTB&GVAR1.2+X'EC'
DC      X'EF'
ORG     DEFTB&GVAR1.2+X'FB'
DC      X'96EDF0'
ORG     DEFTB&GVAR1.2+X'FF'
DC      X'E3'

```

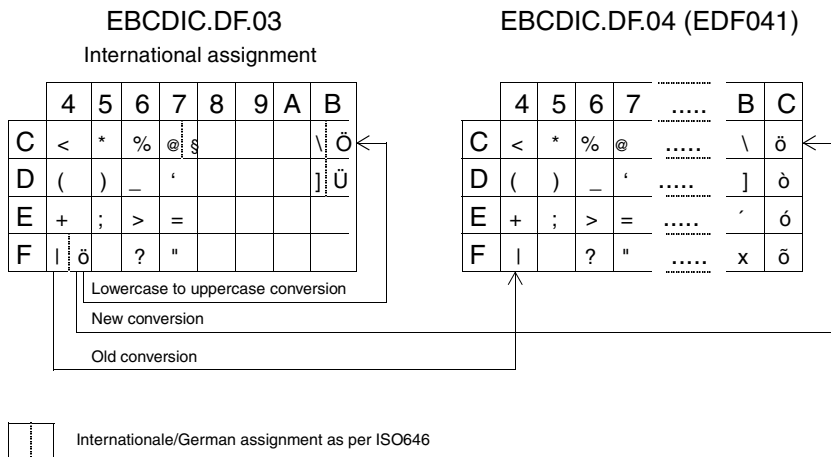
\*

```

*
* Redefinition of the properties table
* X'C1': defines displayable uppercase letter
* X'D1': defines displayable lowercase letter
*
      ORG  DEFTB&GVAR1.3+X'4F'   * ö
      DC   X'D1'
      ORG  DEFTB&GVAR1.3+X'BB'   * Ä Ö Ü
      DC   X'C1C1C1'
      ORG  DEFTB&GVAR1.3+X'FB'   * ä
      DC   X'D1'
      ORG  DEFTB&GVAR1.3+X'FD'   * ü
      DC   X'D1'
      ORG  DEFTB&GVAR1.3+X'FF'   * ß
      DC   X'D1'
*
      MEND
    
```

*Example*

The redefinition of the letter ö for conversion tables and tables for converting lowercase letters to uppercase letters is illustrated in the figure below.



**Source for GNLMTAB with NLSCTAB**

```

GNLMTAB CSECT
*
      AMODE   ANY
      RMODE   ANY
*
      MCALL   EDF041
      MCALL   ISO88591
      MCALL   EDF03IRV
      MCALL   ITOD
      NLSCTAB ((EDF041,1,*REF,*FULL,*EBCDIC,EDF041, ),           ---(1)
              (ISO88591,1,*NORMAL,*FULL,*ISO,ISO88591, ),       ---(2)
              (E03GERM,1,*NORMAL,*RESTR,*EBCDIC,EDF03IRV,ITOD)) ---(3)
      END

```

**(1) Definition of the standard character set EBCDIC.DF.04-1**

The ISO code variant has the number '1'. The code is a family reference code and contains a complete set of all the characters that can be used in the family (which is a necessary condition for a reference code). It is an EBCDIC code, created from the EDF041 standard code and not modified.

**(2) Definition of the standard character set ISO8859-1**

The ISO code variant has the number '1'. The code is not a family reference code but it contains the complete character set of the family reference code. It is an ISO code, created from the ISO88591 code shipped with XHCS, and unmodified.

**(3) Definition of a German version of EBCDIC.DF.03: E03GERM**

The ISO code variant has the number '1'. The code is not a family reference code and does not contain all the characters of the family reference code. It is an EBCDIC code, created from the EDF03IRV standard code, which is redefined using the ITOD macro.

**Source for GNLMTAB with NLSHEAD and NLSCCS**

```

GNLMTAB CSECT
*
      AMODE     ANY
      RMODE     ANY
*
      MCALL     EDF041
      MCALL     ISO88591
      MCALL     EDF03IRV
      MCALL     ITOD
*
      NLSHEAD   NCCS=3
*
      NLSCCS    (EDF041,1,*REF,*FULL,*EBCDIC,EDF041,)          ----(1)
      NLSCCS    (ISO88591,1,*NORMAL,*FULL,*ISO,ISO88591,)      ----(2)
      NLSCCS    (E03GERM,1,*NORMAL,*RESTR,*EBCDIC,EDF03IRV,ITOD) ----(3)
*
      END

```

**(1) Definition of the standard character set EBCDIC.DF.04-1**

The ISO code variant has the number '1'. The code is a family reference code and contains a complete set of all the characters that can be used in the family (which is a necessary condition for a reference code). It is an EBCDIC code, created from the EDF041 standard code and not modified.

**(2) Definition of the standard character set ISO8859-1**

The ISO code variant has the number '1'. The code is not a family reference code but it contains the complete character set of the family reference code. It is an ISO code, created from the ISO88591 code shipped with XHCS, and unmodified.

**(3) Definition of a German version of EBCDIC.DF.03: E03GERM**

The ISO code variant has the number '1'. The code is not a family reference code and does not contain all the characters of the family reference code. It is an EBCDIC code, created from the EDF03IRV standard code, which is redefined using the ITOD macro.

## 5.4.4 Installation of user-defined tables

Systems support can install a user-defined GNLMTAB in a separate library or in the module library SYSLNK.XHCS-SYS.020 by overwriting the originally delivered library. By installing a separate library, systems support can be sure that the individually changed module is not overwritten when the product XHCS is installed again. This functionality is available with the product IMON.

The user-specific library must be linked with XHCS using the following IMON command:

```
SET-INSTALLATION-PATH PATH-NAME=separate-library,  
LOGICAL-IDENTIFIER=SYSLNK.USER,  
INSTALLATION-UNIT=XHCS-SYS(VERSION=V02.0A00)
```

If systems support wants to prevent XHCS using a user-specific library, the following IMON command can be issued:

```
SET-INSTALLATION-PATH PATH-NAME=*NONE,  
LOGICAL-IDENTIFIER=SYSLNK.USER,  
INSTALLATION-UNIT=XHCS-SYS(VERSION=V02.0A00)
```

Further information on product installation can be found in the “IMON (BS2000/OSD)” manual [9].

If no user-specific library exists, XHCS-SYS loads the table module GNLMTAB using the module library SYSLNK.XHCS-SYS.020, as installed by IMON with the logical name SYSLNK.

If XHCS-SYS was not installed with IMON, the GNLMTAB module must be transferred to the module library installed under TSOS, i.e. \$TSOS.SYSLNK.XHCS-SYS.020.

If GNLMTAB is not found in the selected library, XHCS issues message GLN0001 at the console, together with the library name.



Modifications (e.g. to the library containing the tables) do not take effect until the system is loaded again.



## 5.5 Summary of rules and conventions

The various rules and conventions relating to the creation of table modules are summarized below:

- There is only one reference code in each code family. You must select the EBCDIC.DF.04-n code of the code family as the reference code, where -n is the ISO code variant number (hexadecimal).
- The reference code must contain all the characters defined in the other codes of its family. If it does not, there can be no conversion.
- In the conversion table of the reference code, the position of the character must correspond to the character value (i.e. byte no. i must have the character value i).
- Characters undefined in a nonreference code are mapped to NUL (X'00') in the conversion table. If the reference code contains undefined characters (e.g. a code family with the ISO code variant number 7), each undefined character in the nonreference code must be mapped to an undefined character in the reference code.
- Some characters cannot be redefined because they must be identified and processed regardless of which extended character set is used. These are the characters of the EBCDIC kernel, the characters # @ \$ ! and ", and the control characters.
- In the table for converting lowercase letters to uppercase, characters not converted to uppercase letters are mapped to themselves.
- The sort table must be unambiguous. This means that each value in the range from X'00' to X'FF' must occur only once in this table.
- For some software products, fully compatible codes are always 8-bit codes and restricted codes always 7-bit. Even though this does not always apply as far as XHCS is concerned, you should assume that it does when defining new codes.
- Standardized codes have standard names and should not be redefined.
- Please note that if you modify tables, errors will occur during data transfer with other BS2000 computers unless the latter use the same tables. You must also observe the rules and conventions described in this section; for example, that you should not change characters that are significant for the system (e.g. the characters of the command syntax). To avoid errors of this kind, you should use the extended character sets shipped as standard, and regard table modification as an additional XHCS option to be used only in special cases (e.g. for reasons of compatibility).

## 5.6 Defining additional Unicode characters

XHCS V2.0 enables you to define additional Unicode characters. This is important because XHCS initially supplies only a limited code set. The variants concerned here are the ISO8859 variants 1, 2, 3, 4, 5, 7, 9 and 15, as well as additional characters which are used by our customers. If you want to support further characters for Unicode in addition to these, please proceed as described below:

First check whether the extension contains characters which are included in one of the ISO8859 variants which exist but are not yet supported. If this is the case, this variant can be supported by a local macro in the GNLMTAB module, as described in the [section “Creation and modification of code tables” on page 119](#). Note here that the 8-bit sort table has no meaning for Unicode. The corresponding sort information for Unicode must be defined separately, see [page 131](#).

### 5.6.1 Defining pseudo 8-bit code

If you want to add characters which are not included in any ISO8859 variant, you must initially define a pseudo 8-bit code.

- ▶ To do this, use either the ISOEXT2 or ISOEXT3 macro. ISOEXT1 is already used for defining additional characters required by our customers.
- ▶ First of all sort the Unicode characters on a binary basis.
- ▶ Assign each character an 8-bit position, beginning with mit 0xA0. Positions 0x00 through 0x9F are the same in all ISO8859 variants and therefore cannot be used.
- ▶ Transfer this assignment to the table 8-bit code -> Unicode position, which is the last table of the ISOEXT2(3) macro.
- ▶ Use this table to define the lowercase/uppercase table and the properties table. The sort table has no meaning at Unicode level (see [page 131](#)).
- ▶ Enter values for the conversion table and its inverse table with the same mapping at the positions which have been assigned. Everything else is 0x00.
- ▶ Anchor the pseudo code in the GNLMTAB module:  

```
NLSCCS (ISOEXT2|3 , 0 , *NORMAL , *FULL , *ISO , ISOEXT2|3,)
```
- ▶ Increment the counter of NLSHEAD.

The pseudo code thus defined is only used in helping to extend the character set supported at Unicode level. It has no meaning at 8-bit level. The lowercase/uppercase table, its inverse table, and the properties table for Unicode are calculated dynamically from the existing 8-bit codes.

## 5.6.2 Defining sort information

The sort information for Unicode is stored in the GNLMTAB module, directly behind the macros for defining the 8-bit code tables. Its structure is the same as that described under NLSCOD TABLE=SORT on [page 67](#) and initially consists of X'3000' = 12288 entries (collation entries) of 8 bytes each for possible Unicode positions between 0 and 2FFF. The number of an entry corresponds to its Unicode position. Currently approx. 700 positions are assigned. The entries are assigned using the first collation element of a collation entry and, if required, a reference to a second collation element. The range of the second and third collation elements begins after these X'3000'\*8 = X'18000' bytes. This is X'2000' bytes in size and therefore has space for 1024 (= X'2000' / 8 = X'400') collation elements. Detailed information on this is provided under NLSCOD, TABLE=SORT on [page 67](#).

## 5.6.3 Assembling GNLMTAB and linking in an extended code base

- ▶ After you have defined the sort information, assemble the GNLMTAB module.
- ▶ Link the extended code base when you start the XHCS-SYS subsystem or dynamically using the /ADD-CODE-TABLES command (see also [page 132](#)).

## 5.7 Adding new coded character sets dynamically: ADD-CODE-TABLES

The ADD-CODE-TABLES command allows to add dynamically new coded character sets, without reloading the system.

ADD-CODE-TABLES
-----------------

FROM-LIBRARY = filename_1..54_without-generation
--

### Operand description

#### **FROM-LIBRARY = filename\_1..54\_without-generation**

XHCS-SYS will use this file to dynamically add new variants by using the module GNLMTAB (R type or L type) that must be present in the library.

### Programming notes

XHCS-SYS will automatically add new variants and display them to inform the system administrator. These new variants, once given, are available to any running tasks, applications.

If variants defined in the added module are already known (from start-up or from a preceding /ADD-CODE-TABLES command), they won't be taken into account to ensure the tables consistency (i.e. previous definitions are kept).

The command can never be used to modify existing tables.

### Privileges required to issue the command

The privileges of the XHCS-SYS command are defined in the following table :

Command	Privileges
ADD-CODE-TABLES	TSOS, OPERATING

**Short description of the different command return codes**

	<b>Return code</b>	<b>Meaning</b>	<b>List of possible XHCS-SYS messages</b>
1	00 00 CMD0001	Command accepted	Information messages: GNL0006 GNL0007
2	02 00 GNL0100	Warning during command processing	Information messages: GNL0003
3	00 40 CMD0216	User is not authorized to issue this command. Command rejected	Error messages: GNL0011
4	00 20 GNL0101	Error during external system call. Command not processed	Error messages: GNL0001 GNL0004
5	00 40 GNL0102	Unrecoverable error during processing. Command not executed	Error messages : GNL0008
6	00 80 GNL0005	Command not possible for a short period of time. Retry later	Retry messages: GNL0005

**Possible error and warning cases***Errors*

- As the module will be bound by a call to BLS interfaces (\$PBBND1), all the potential binding errors will be given by BLS itself.
- Once the module GNLMTAB bound, XHCS may find that it has an invalid format or an invalid version. In this case, the command processing is rejected.

*Warnings*

- No new variant exists in the module GNLMTAB bound from the given library.

## 5.8 Arabic and Persian codes

BS2000/OSD-BC uses VTSU as of V11.0 and special routines to support bidirectional alphabets.

### 5.8.1 Supported codes

XHCS supports the following codes for bidirectional alphabets:

ISO code variant	XHCS name (CCSN)	Complete name	Representation
6	EDF046 EEHCLAA	EBCDIC.DF.04-6.ARA EBCDIC.EHC.LA.ARA	Latin/Arabic with Arabic digits
A	EDF04A EEHCLAI	EBCDIC.DF.04-6.IND EBCDIC.EHC.LA.IND	Latin/Arabic with Indian digits
C	EDF04C EEHCLFI	EBCDIC.DF.04-FAR.INT EBCDIC.EHC.LF.INT	Latin/Farsi with international digits
D	EDF04D EEHCLFF	EBCDIC.DF.04-FAR.FAR EBCDIC.EHC.LF.FAR	Latin/Farsi with Persian digits
E	EDF04E EEHCNAA	EBCDIC.DF.04-NAF.ARA EBCDIC.EHC.NA.ARA	French/Arabic with Arabic digits
F	EDF04F EEHCNAI	EBCDIC.DF.04-NAF.IND EBCDIC.EHC.NA.IND	French/Arabic with Indian digits

Unlike other character sets, the terminal code (line code) for bidirectional alphabets is not fully compatible with the computer code used. A few special characters appear only once in terminal code, regardless of whether or not they come from the Latin or Arabic alphabet.

On the computer side (EBCDIC code) these special characters appear twice, so there is, for instance, two “+” characters: one is assigned to the Latin alphabet, while the other is assigned to the Arabic. This makes it possible to uniquely determine Latin or Arabic membership.

Depending on the difference in the terminal and computer code structure, these codes must be edited on the basis of VTSU as of V11.0.



Only the CCS for Latin/Arabic is defined in ISO 8859 (variant 6).

Unlike other character sets, the character sets that support bidirectional alphabets are identified by six “pseudo ISO variant numbers”. The number that identifies this code family is not an existing ISO variant number.

### **5.8.2 Rules for Arabic codes**

Two terminal codes that cover the same alphabet are equal as far as XHCS is concerned if they differ only in how they display digits. As a result, the contents of code families 6 and A, C and D, E and F should be identical when they are defined. To achieve this, create the same default codes and redefinition macros for equivalent codes (“deftab-” and “macroname-” operands of the NLSCTAB and NLSCCS macros). This ensures that the contents of the tables are identical. This assignment of codes is also displayed in the names of Arabic default codes which differ only in the last letter.

## 5.9 Support of the euro symbol in XHCS

Support for the euro symbol € in XHCS is provided in two ways. You can

- continue to use the ISO code variants 8859-1, 8859-2 and 8859-9, or
- introduce the new ISO code variant 8859-15.

Both methods are described in this section.



- Introducing the new variant ISO 8859-15 leads to incompatibilities regarding support for Arabic codes. This is described in [section “Compatibility restrictions with regard to Arabic code variant F” on page 138](#).
- The euro symbol is not supported in Cyrillic and ISO 7-bit codes.

### 5.9.1 Use of ISO code variants ISO 8859-1/-2/-7/-9

To support the euro symbol €, you can continue to use the ISO code variants 8859-1 (ISO88591), 8859-2 (ISO88592), 8859-7 (ISO88597) and 8859-9 (ISO88599) and the associated EBCDIC codes (see table on [page 144](#)).

Regardless of the Latin 8-bit character code, emulations can interpret the character at position X'A4' (ASCII) or X'9F' (EBCDIC) either as the currency symbol ¤ as before, or as the new euro symbol. The relevant interpretation is selected via an option in the emulation menu or via an entry in the configuration file (.INI) of the emulation. Simultaneous output of the currency symbols ¤ and the euro symbol is not possible.

The value set for the option (“currency symbol“ or “euro symbol“) is valid for all three code variants. This means, for example, that it is not possible to set the option for 8859-1 to “euro symbol“ and for the two other variants to “currency symbol“.

Using this procedure means that no new, differently named character sets are required for the euro symbol. Files may contain the euro symbol without their being assigned a new code attribute (Coded Character Set Name). Existing applications do not need to be modified. The modified interpretation of the old code tables is compatible with the old interpretation. Character attributes, conversions and sorts are not affected.



## 5.9.2 Introduction of the ISO code variant ISO 8859-15

XHCS supports the new norm variant ISO 8859-15 (Latin-9) through a new code group with the ISO tables and the corresponding EBCDIC equivalent. In this case, the euro symbol € is situated at position X'A4' and replaces the currency symbol ¤, which occupies this position in the other Latin ISO variants.

In addition, special characters have been replaced by European special letters in this code variant compared with ISO 8859-1. This means that Latin-9 supports all euro-relevant languages, albeit with the restriction that character attributes that can be queried via program interfaces are changed and become incompatible.

The code group ISO 8859-15 contains the tables for the codes

- EDF04F (EBCDIC, reference code of the group)
- ISO8859F (ASCII)
- WCP1252P (partially compatible variant of Windows character set 1252, which contains the euro symbol).

Specifying this code group for the file attribute CODED-CHARACTER-SET-NAME explicitly defines for text files that all X'9F' EBCDIC codes they contain must be interpreted as euro symbols.

### 5.9.2.1 Compatibility restrictions with regard to Arabic code variant F

XHCS already supports a code variant with the designation 'F' for Arabic codes which is not a standard ISO variant (see [section "Arabic and Persian codes" on page 134](#)). This results in a conflict with the same designation in the standard ISO variante 8859-15. Simultaneous use of the Arabic variant 'F' and the ISO variant 'F' with the euro symbol € is not possible.

The European variant ISO 8859-15 has been activated in the standard installation.

Support for the Arabic variant means that VTSU special routines have to be installed (see [section "Support of special terminals via VTSU special routines" on page 35](#)).

- If these routines have been installed, VTSU regards variant 'F' as the Arabic variant.
- If these routines have not been installed, VTSU interprets variant 'F' as the European variant.

Systems support is responsible for ensuring no conflicts arise during installation.

---

## 6 Preparations for use

This chapter describes the installation steps necessary to implement an operational 8-bit terminal or a national 7-bit data display terminal.

### 6.1 Preparations for XHCS support

#### 6.1.1 PDN generation

Type 9758, 9759 and 9763 data display terminals must be generated with STATTYP= DSS-9763 and type 9756 data display terminals with STATTYP= DSS-9755 in PDN. 8-bit printers must be defined in PDN using their corresponding terminal types. In addition, printers must be defined as 8-bit printers by means of the PDN free text parameter (PDN as of V11) or in the VTSU operating parameter file. VTSU assumes that the correct character set is set for the printer.



The PDN free text parameter has a higher priority than the corresponding VTSU operating parameters.

## 6.1.2 XHCS

The optional product XHCS is a dynamic subsystem loaded via DSSM; it cannot be unloaded during operation. By default, XHCS supports ISO code variants 1, 2, 3, 4, 5, 7, 9 and F. The corresponding code tables are defined in the GNLMTAB module (see [section “Components and program interfaces” on page 28](#)).

In addition, XHCS supports the corresponding standard EBCDIC codes for all other code variants (see [page 143](#)). These codes are predefined in the GNLMTAB module but they are deactivated by means of comment characters. By removing the corresponding comment characters, the required code tables can be activated. The GNLMTAB module must be assembled and the translated module must be added to the SYSLNK.XHCS-SYS.020 module library. The modified GNLMTAB module can be linked into the system using the /ADD-CODE-TABLES command (see the [section “Adding new coded character sets dynamically: ADD-CODE-TABLES” on page 132](#)). The modification does not become effective until the next time the XHCS-SYS subsystem is loaded, i.e. after the next system start.

Additional tables can be defined by modifying the tables contained in the GNLMTAB module. This allows you to customize your own EBCDIC variants.

## 6.1.3 Activating the 8-bit environment for 8-bit data display terminals

8-bit mode can either be permanently set (permanent 8-bit mode) or activated only for a specific program.

### Permanent 8-bit mode

Permanent 8-bit mode can be set **systemwide** for TIAM, DCAM and openUTM applications by using the VTSU operating parameter file; the name of the user default character set (CCSN) is automatically activated for the 8-bit data display terminal. In addition, users can activate this CCSN for TIAM applications by using the following TIAM command:

```
/MODIFY-TERMINAL-OPTIONS CODED-CHARACTER-SET=8-BIT-DEFAULT
```

### Program-related 8-bit mode

Programs (TIAM and DCAM applications) can set 8-bit mode via the VTSUCB.

8-bit mode is also automatically activated for certain utility programs (e.g. EDT, IFG/FHS, etc.) if you use files or formats that have the file identification CCSN that is supported by the terminal.

*Example*

The 9759-M2 terminal is set to the Greek character set in the SIDATA keyboard setting. 8-bit mode has not been activated. 8-bit mode is automatically activated for the EDT editor when the EDT reads in a file with a Greek CCSN.

## 6.2 Preparation for VTSU special routines

### 6.2.1 PDN generation

Type 9758 and 9763 terminals must be generated with STATYP= DSS-9763, type 9756 terminals with STATYP=DSS-9755, and national 7-bit terminals with STATYP = DSS-9750 in PDN. ESC printers must be defined in PDN using their corresponding terminal types. In addition, printers must be defined as 8-bit printers by means of the PDN free text parameter (PDN as of V11) or in the VTSU operating parameter file (see the “VTSU“ manual [1]). VTSU assumes that the correct character set is set for the printer.



The PDN free text parameter has a higher priority than the corresponding VTSU operating parameters.

### 6.2.2 VTSU

After the configuration file is modified and/or after the VTSU special routines are installed, the VTSU subsystem must be unloaded and the system restarted.

### 6.2.3 Activating the 8-bit environment for Arabic/Persian terminals

You can only use Arabic/Persian 8-bit data display terminals if you have requested support for these data display terminals during the installation procedure. The installation procedure is described in the “VTSU“ manual [1]. Activation of 8-bit mode is described on [page 140](#). A list of valid CCSNs for ARA/FAR is provided on [page 144](#).

### 6.2.4 Activating European 7-bit data display terminals

You can only use European 7-bit data display terminals if you have requested support for these data display terminals during the installation procedure. The installation procedure is described in the “VTSU“ manual [1]. In addition, European 7-bit data display terminals have to be defined in a configuration file. The creation of the configuration file is likewise described in the “VTSU“ manual [1].

## 6.2.5 Activating ESC printers

ESC printers have to be defined in a configuration file. Users are responsible for creating the configuration file; the procedure for this is described in the “VTSU“ manual [1]. In addition, printers must be defined as 8-bit printers by means of the PDN free text parameter (PDN as of V11) or in the VTSU operating parameter file (see the “VTSU“ manual [1]). VTSU assumes that the correct character set is set for the printer.

---

# 7 Tables

## 7.1 Standard CCSN tables

### 7-bit line codes (ISO codes) and associated EBCDIC codes

ISO code variant	XHCS name	Complete name	Representation
1	ISO646 EDF03IRV EDF03DRV	ISO646-IRV EBCDIC.DF.03-IRV *) EBCDIC.DF.03-DRV	International 7-bit code

## 8-bit line codes (ISO codes) and associated EBCDIC codes

ISO code variant	XHCS name (CCSN)	Complete name	Representation
1	ISO88591 EDF041 EDF04DRV	ISO8859-1 EBCDIC.DF.04(-1) *) EBCDIC.DF.04(-1) DRV	Latin alphabet no. 1  EDF03DRV extension
2	ISO88592 EDF042 EEHCL2	ISO8859-2 EBCDIC.DF.04-2 EBCDIC.EHC.L2 *)	Latin alphabet no. 2
3	ISO88593 EDF043	ISO8859-3 EBCDIC.DF.04-3	Latin alphabet no. 3
4	ISO88594 EDF044	ISO8859-4 EBCDIC.DF.04-4	Latin alphabet no. 4
5	ISO88595 EDF045 EEHCLC EEHCLC1	ISO 8859-5 EBCDIC.DF.04-5 EBCDIC.EHC.LC *) EBCDIC.EHC-LC.1 **)	Latin/Cyrillic alphabet
6	EDF046 EEHCLAA	EBCDIC.DF.04-6.ARA EBCDIC.EHC.LA.ARA*)	Latin/Arabic alphabet with Arabic digits
A	EDF04A EEHCLAI	EBCDIC.DF.04-6.IND EBCDIC.EHC.LA.IND*)	Latin/Arabic alphabet with Indian digits
B	EDF04B	EBCDIC.DF.04.BIB.9756	Character set for 9756 library terminal according to DIN 66003/ DIN 31624
C	EDF04C EEHCLFI	EBCDIC.DF.04-FAR.INT EBCDIC.EHC.LF.INT*)	Latin/Farsi alphabet with international digits
D	EDF04D EEHCLFF	EBCDIC.DF.04-FAR.FAR EBCDIC.EHC.LF.FAR	Latin/Farsi alphabet with Persian digits
E	EDF04E EEHCNAA	EBCDIC.DF.04-NAF.ARA EBCDIC.EHC.NA.ARA*)	French/Arabic alphabet with Arabic digits
F	EDF04F EEHCNAI	EBCDIC.DF.04-NAF.IND EBCDIC.EHC.NA.IND*)	French/Arabic alphabet with Indian digits



ISO code variant	XHCS name (CCSN)	Complete name	Representation
7	ISO88597 EDF047 EEHCLG	ISO8859-7(ELOT928) EBCDIC.DF.04-7 EBCDIC.EHC.LG *)	Latin/Greek alphabet
9	ISO88599 EDF049  or  EDF049BE	ISO 8859-9 EBCDIC.DF.04-9 *)   EBCDIC.DF.04.BIB	Latin alphabet no. 5   Library character set according to DIN 31628/2 (emulation)
F	ISO8859F EDF04F WCP1252P	ISO8859-15 EBCDIC.DF.04-F Windows Code Page 1252 Partial	Latin alphabet no. 9

\*) Recommended standard EBCDIC

\*\*\*) Recommended standard EBCDIC for new customers

The names of the various ISO 8859 and EBCDIC.DF.04 codes are always represented using 'ISO 8859x' and 'EDF04x' respectively. This rule applies to all ISO code variant numbers.

## 7.2 Overview of XHCS VTSUCB return information

The following table provides you with an overview of the VTSUCB return information that applies to XHCS. The return information issued depends on whether or not XHCS is loaded, what user/system default code or terminal type you are using and what parameters have been defined in the VTSU control block.

XHCS	8-bit code*	8-bit terminal	VTSUCB parameter CODETR   CCSNAME		Return code*	Notes
N	—	—	Y	—	00000000	5)
N	—	—	—	Y	60010004	
Y	—	N	Y	—	00000000	5)
Y	—	N	—	Y	61010004	
Y	N	Y	Y	—	00000000	5)
Y	N	Y	—	Y	61010004	
Y	Y	Y	Y	—	00000000	1)
Y	Y	Y	—	*EXTEND	00000000	3)
Y	Y	Y	—	EDF03IRV	00000000	2)
Y	Y	Y	—	other	00000000	4)
Y	Y	Y	—	other	1E010004	4)
Y	Y	Y	—	other	86010004	4)

— not relevant

Y provided

N not provided

\* N means either that the user/system default code is not a full 8-bit code or that this code is not supported by the current terminal.

\* Return codes with no entry in the Notes column are described in the “VTSU” [1] manual.

1) The CODETR parameter is ignored until a message is sent in 8-bit mode.

2) The message is sent in 7-bit mode. The message is processed as if no VTSU control block were being used or no code name had been defined.

3) The message sent uses the user default code.

4) The message sent uses the defined code. This code must be known in XHCS and supported by the terminals being used. Otherwise the message is rejected with the return code X'1E' (name unknown in XHCS) or X'86' (name is not supported by terminal).

5) The CODETR parameter is ignored. Environment errors are detected only via the CCSNAME parameter.

## 7.3 Supported line codes and BS2000 EBCDIC codes

### 7.3.1 Supported line codes and BS2000 EBCDIC codes for European alphabets

Key:



Control character cannot be assigned



Position cannot be assigned




EBCDIC.DF03 kernel

## 7.3.1.1 8-bit line codes and associated EBCDIC codes for European alphabets


Latin alphabet no. 1 ISO 8859-1 (CCSN: ISO88591)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00			SP	0	@	P	`	p			NBSP	°	À	Ð	à	ð
01			!	1	A	Q	a	q			ı	±	Á	Ñ	á	ñ
02			"	2	B	R	b	r			ϕ	²	Â	Ò	â	ò
03			#	3	C	S	c	s			£	³	Ã	Ó	ã	ó
04			\$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô
05			%	5	E	U	e	u			¥	µ	Å	Õ	å	õ
06			&	6	F	V	f	v			¦	¶	Æ	Ö	æ	ö
07			'	7	G	W	g	w			§	·	Ç	×	ç	÷
08			(	8	H	X	h	x			¨	¸	È	Ø	è	ø
09			)	9	I	Y	i	y			©	¹	É	Ù	é	ù
0A			*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú
0B			+	;	K	[	k	{			«	»	Ë	Û	ë	û
0C			,	<	L	\	l				¬	¼	Ì	Ü	ì	ü
0D			-	=	M	]	m	}			SHY	½	Í	Ý	í	ý
0E			.	>	N	^	n	~			®	¾	Î	Þ	î	þ
0F			/	?	O	_	o				¯	¿	Ï	ß	ï	ÿ

Extended BS2000 code **EBCDIC.DF.04-1** (CCSN: **EDF041**)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-	ø	Ø	°	μ	φ	ù	ı	Ù	0
01					NBSP	é	/	É	a	j	—	£	A	J	÷	1
02					â	ê	Â	Ê	b	k	s	¥	B	K	S	2
03					ä	ë	Ä	Ë	c	l	t	·	C	L	T	3
04					à	è	À	È	d	m	u	©	D	M	U	4
05					á	í	Á	Í	e	n	v	§	E	N	V	5
06					ã	î	Ã	Î	f	o	w	¶	F	O	W	6
07					å	ï	Å	Ï	g	p	x	¼	G	P	X	7
08					ç	ì	Ç	Ì	h	q	y	½	H	Q	Y	8
09					ñ	ß	Ñ	¨	i	r	z	¾	I	R	Z	9
0A					`	!	^	:	«	ª	ı	¬	SHY	¹	²	³
0B					.	\$	,	#	»	º	¿	[	ô	û	Ô	{
0C					<	*	%	@	ð	æ	Ð	\	ö	ü	Ö	Ü
0D					(	)	_	'	ý	,	Ý	]	ò	Û	Ò	}
0E					+	;	>	=	þ	Æ	Þ	´	ó	ú	Ó	Ú
0F							?	"	±	¤	®	×	õ	ÿ	Õ	~


Extended BS2000 code **EBCDIC.DF.04-DRV** (CCSN: **EDF04DRV**)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-	ø	Ø	°	μ	φ	{	}	\	0
01					NBSP	é	/	É	a	j	~	£	A	J	÷	1
02					â	ê	Â	Ê	b	k	s	¥	B	K	S	2
03						ë	ï	Ë	c	l	t	·	C	L	T	3
04					à	è	À	È	d	m	u	©	D	M	U	4
05					á	í	Á	Í	e	n	v	@	E	N	V	5
06					ã	î	Ã	Î	f	o	w	¶	F	O	W	6
07					å	ï	Å	Ï	g	p	x	¼	G	P	X	7
08					ç	ì	Ç	Ì	h	q	y	½	H	Q	Y	8
09					ñ		Ñ	¨	i	r	z	¾	I	R	Z	9
0A					`	!	^	:	«	ª	¡	¬	SHY	¹	²	³
0B					.	\$	,	#	»	º	¿	Ä	ô	û	Ô	ä
0C					<	*	%	§	ð	æ	Ð	Ö	[	]	Û	Ù
0D					(	)	_	'	ý	¸	Ý	Ü	ò	ù	Ò	ü
0E					+	;	>	=	þ	Æ	Þ	´	ó	ú	Ó	Ú
0F					ö	–	?	"	±	¤	®	×	õ	ÿ	Ö	ß

## Latin alphabet no. 9 ISO 8859-15 (CCSN: ISO885915)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00			SP	0	@	P	`	p			NBSP	°	À	Ð	à	ð
01			!	1	A	Q	a	q			ı	±	Á	Ñ	á	ñ
02			"	2	B	R	b	r			ø	²	Â	Ò	â	ò
03			#	3	C	S	c	s			£	³	Ã	Ó	ã	ó
04			\$	4	D	T	d	t			€	Ž	Ä	Ô	ä	ô
05			%	5	E	U	e	u			¥	μ	Å	Ö	å	ö
06			&	6	F	V	f	v			Š	¶	Æ	Ö	æ	ö
07			'	7	G	W	g	w			§	·	Ç	×	ç	÷
08			(	8	H	X	h	x			š	ž	È	Ø	è	ø
09			)	9	I	Y	i	y			©	¹	É	Ù	é	ù
0A			*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú
0B			+	;	K	[	k	{			«	»	Ë	Û	ë	û
0C			,	<	L	\	l				¬	œ	Ì	Ü	ì	ü
0D			-	=	M	]	m	}			SHY	œ	Í	Ý	í	ý
0E			.	>	N	^	n	~			®	ÿ	Î	Þ	î	þ
0F			/	?	O	_	o				—	¿	Ï	ß	ï	ÿ

Extended BS2000 code **EBCDIC.DF.04-F** (CCSN: **EDF04F**)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-	ø	Ø	°	μ	φ	ù	Š	Ù	0
01					NBSP	é	/	É	a	j	—	£	A	J	÷	1
02					â	ê	Â	Ê	b	k	s	¥	B	K	S	2
03					ä	ë	Ä	Ë	c	l	t	·	C	L	T	3
04					à	è	À	È	d	m	u	©	D	M	U	4
05					á	í	Á	Í	e	n	v	§	E	N	V	5
06					ã	î	Ã	Î	f	o	w	¶	F	O	W	6
07					å	ï	Å	Ï	g	p	x	œ	G	P	X	7
08					ç	ì	Ç	Ì	h	q	y	æ	H	Q	Y	8
09					ñ	ß	Ñ	Š	i	r	z	Ÿ	I	R	Z	9
0A					`	!	^	:	«	ª	¡	¬	SHY	¹	²	³
0B					.	\$	,	#	»	º	¿	[	ô	û	Ô	{
0C					<	*	%	@	ð	æ	Ð	\	ö	ü	Ö	Ü
0D					(	)	_	'	ý	ž	Ý	]	ò	Û	Ò	}
0E					+	;	>	=	þ	Æ	Þ	Ž	ó	ú	Ó	Ú
0F							?	"	±	€	®	×	õ	ÿ	Õ	~




Windows Code Page 1252 Partial (CCSN: **WCP1252P**)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
<b>00</b>			SP	0	@	P	`	p	€		NBSP	°	À	Ð	à	ð
<b>01</b>			!	1	A	Q	a	q			ı	±	Á	Ñ	á	ñ
<b>02</b>			"	2	B	R	b	r			ϕ	²	Â	Ò	â	ò
<b>03</b>			#	3	C	S	c	s			£	³	Ã	Ó	ã	ó
<b>04</b>			\$	4	D	T	d	t					Ä	Ô	ä	ô
<b>05</b>			%	5	E	U	e	u			¥	μ	Å	Õ	å	õ
<b>06</b>			&	6	F	V	f	v				¶	Æ	Ö	æ	ö
<b>07</b>			'	7	G	W	g	w			§	·	Ç	×	ç	÷
<b>08</b>			(	8	H	X	h	x					È	Ø	è	ø
<b>09</b>			)	9	I	Y	i	y			©	¹	É	Ù	é	ù
<b>0A</b>			*	:	J	Z	j	z	Š	š	ª	º	Ê	Ú	ê	ú
<b>0B</b>			+	;	K	[	k	{			«	»	Ë	Û	ë	û
<b>0C</b>			,	<	L	\	l		Œ	œ	¬		Ì	Ü	ì	ü
<b>0D</b>			-	=	M	]	m	}			SHY		Í	Ý	í	ý
<b>0E</b>			.	>	N	^	n	~			®		Î	Þ	î	þ
<b>0F</b>			/	?	O	_	o	DEL		ÿ	¯	¿	Ï	ß	ï	ÿ

## Latin alphabet no. 5 ISO 8859-9 (CCSN: ISO88599)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00			SP	0	@	P	`	p			NBSP	°	À	Ǧ	à	ǧ
01			!	1	A	Q	a	q			ı	±	Á	Ñ	á	ñ
02			"	2	B	R	b	r			ϕ	²	Â	Ò	â	ò
03			#	3	C	S	c	s			£	³	Ã	Ó	ã	ó
04			\$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô
05			%	5	E	U	e	u			¥	µ	Å	Õ	å	õ
06			&	6	F	V	f	v			ı	¶	Æ	Ö	æ	ö
07			'	7	G	W	g	w			§	·	Ç	×	ç	÷
08			(	8	H	X	h	x			¨	¸	È	Ø	è	ø
09			)	9	I	Y	i	y			©	¹	É	Ù	é	ù
0A			*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú
0B			+	;	K	[	k	{			«	»	Ë	Û	ë	û
0C			,	<	L	\	l				¬	¼	Ì	Ü	ì	ü
0D			-	=	M	]	m	}			SHY	½	Í	İ	í	ı
0E			.	>	N	^	n	~			®	¾	Î	Ş	î	ş
0F			/	?	O	_	o				—	¿	Ï	ß	ï	ÿ


Extended BS2000 code **EBCDIC.DF.04-9** (CCSN: **EDF049**)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-	ø	Ø	°	μ	¢	ù	ı	Ù	0
01					NBSP	é	/	É	a	j	—	£	A	J	÷	1
02					â	ê	Â	Ê	b	k	s	¥	B	K	S	2
03					ä	ë	Ä	Ë	c	l	t	·	C	L	T	3
04					à	è	À	È	d	m	u	©	D	M	U	4
05					á	í	Á	Í	e	n	v	§	E	N	V	5
06					ã	î	Ã	Î	f	o	w	¶	F	O	W	6
07					å	ï	Å	Ï	g	p	x	¼	G	P	X	7
08					ç	ì	Ç	Ì	h	q	y	½	H	Q	Y	8
09					ñ	ß	Ñ	¨	i	r	z	¾	I	R	Z	9
0A					`	!	^	:	«	ª	ı	¬	SHY	¹	²	³
0B					.	\$	,	#	»	º	ı	[	ô	û	Ô	{
0C					<	*	%	@	ǧ	æ	Ĝ	\	ö	ü	Ö	Ü
0D					(	)	_	'	ı	˙	ı	]	ò	Û	Ò	}
0E					+	;	>	=	§	Æ	§	'	ó	ú	Ó	Ú
0F							?	"	±	¤	®	×	õ	ÿ	Õ	~


## Latin alphabet no. 2 ISO 8859-2 (CCSN: ISO88592)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00			SP	0	@	P	`	p			NBSP	°	Ř	Đ	ř	ď
01			!	1	A	Q	a	q			Ą	ą	Á	Ń	á	ń
02			"	2	B	R	b	r			˘	˙	Â	Ň	â	ň
03			#	3	C	S	c	s			Ł	ł	Ǻ	Ó	ǻ	ó
04			\$	4	D	T	d	t			Ɑ	´	Ä	Ô	ä	ô
05			%	5	E	U	e	u			Ł	ł	Ł	Ů	í	ů
06			&	6	F	V	f	v			Ś	ś	Ć	Ö	ć	ö
07			'	7	G	W	g	w			Ş	˘	Ç	×	ç	÷
08			(	8	H	X	h	x			¨	˙	Č	Ř	č	ř
09			)	9	I	Y	i	y			Š	š	É	Ů	é	ů
0A			*	:	J	Z	j	z			Ş	ş	Ę	Ú	ę	ú
0B			+	;	K	[	k	{			ř	ř	Ě	Ú	ě	ů
0C			,	<	L	\	l				Ž	ž	Ě	Ü	ě	ü
0D			-	=	M	]	m	}			SHY	˘	Í	Ý	í	ý
0E			.	>	N	^	n	~			Ž	ž	Î	Ţ	î	ţ
0F			/	?	O	_	o				Ž	ž	Ď	ß	ď	·

Extended BS2000 code **EBCDIC.DF.04-2** (CCSN: **EDF042**)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-	ř	Ř	°	ı	˘	ú	Ś	Ů	0
01					NBSP	é	/	É	a	j	Ž	ł	A	J	÷	1
02					â	ę	Â	Ę	b	k	s	Ł	B	K	S	2
03					ä	ë	Ä	Ë	c	l	t	˘	C	L	T	3
04					ř	č	Ř	Č	d	m	u	Š	D	M	U	4
05					á	í	Á	Í	e	n	v	Ş	E	N	V	5
06					ă	î	Ă	Î	f	o	w	ś	F	O	W	6
07					í	ď	Í	Ď	g	p	x	ż	G	P	X	7
08					ç	ě	Ç	Ě	h	q	y	”	H	Q	Y	8
09					ñ	ß	Ñ	”	i	r	z	ž	I	R	Z	9
0A					`	!	^	:	ř	ş	Ą	Ż	SHY	š	,	ł
0B					.	\$	,	#	ť	ş	ż	[	ô	ű	Ô	{
0C					<	*	%	@	ď	ć	Đ	\	ö	ü	Ö	Ü
0D					(	)	_	'	ý	,	Ý	]	ň	Ű	Ń	}
0E					+	;	>	=	ı	Ć	Ŧ	'	ó	ú	Ó	Ú
0F							?	"	ą	ą	Ž	x	ó	·	Ö	~

Extended BS2000 code **EBCDIC.EHC.L2** (CCSN: **EEHCL2**)

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00					SP	&	-	ř	Ř	°	ˇ	ˇ	ů	“	Ů	0
01					NBSP	ę	/	á	a	j	ß	Á	A	J	÷	1
02					â	Ę	Â	é	b	k	s	É	B	K	S	2
03					ä	ë	Ä	í	c	l	t	Í	C	L	T	3
04					ǎ	Ë	Ǻ	ó	d	m	u	Ó	D	M	U	4
05					ą	ě	Ą	ö	e	n	v	Ö	E	N	V	5
06					ć	î	Ć	ő	f	o	w	Ő	F	O	W	6
07					č	Ě	Č	ú	g	p	x	Ú	G	P	X	7
08					ç	î	Ç	ü	h	q	y	Ü	H	Q	Y	8
09					x	§	ˆ	ú	i	r	z	Ů	I	R	Z	9
0A					`	!	^	:	ł	ł	ł	ł	SHY	í	·	Ł
0B					.	\$	,	#	đ	ł	Đ	[	ô	ş	Ô	{
0C					<	*	%	@	đ	ř	Đ	\	ń	ż	Ń	Ż
0D					(	)	_	'	ý	,	Ý	]	ň	ş	Ń	}
0E					+	;	>	=	ł	Ř	Ť	'	ś	ż	Ś	Ž
0F							?	"	ł	ř	ř	ž	ś	Ž	Ś	~

## Latin alphabet no. 3 ISO 8859-3 (CCSN: ISO88593)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00			SP	0	@	P	`	p			NBSP	°	À		à	
01			!	1	A	Q	a	q			Ĥ	ĥ	Á	Ñ	á	ñ
02			"	2	B	R	b	r			˘	²	Â	Ò	â	ò
03			#	3	C	S	c	s			£	³		Ó		ó
04			\$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô
05			%	5	E	U	e	u				µ	Č	Ĝ	č	ĝ
06			&	6	F	V	f	v			Ĥ	ĥ	Č	Ö	č	ö
07			'	7	G	W	g	w			§	·	Ç	×	ç	÷
08			(	8	H	X	h	x			¨	¸	È	Ĝ	è	ĝ
09			)	9	I	Y	i	y			ı	ı	É	Ù	é	ù
0A			*	:	J	Z	j	z			Ş	ş	Ê	Ú	ê	ú
0B			+	;	K	[	k	{			Ğ	ğ	Ë	Û	ë	û
0C			,	<	L	\	l				Ĵ	ĵ	Ì	Ü	ì	ü
0D			-	=	M	]	m	}			SHY	½	Í	Ů	í	ů
0E			.	>	N	^	n	~					Î	Š	î	š
0F			/	?	O	_	o				Ž	ž	Ï	ß	ï	·

Extended BS2000 code **EBCDIC.DF.04-3** (CCSN: **EDF043**)


	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-	ô	Ô	°	μ	˘	ù	Û	Ù	0
01					NBSP	é	/	É	a	j	Ž	£	A	J	÷	1
02					â	ê	Â	Ê	b	k	s		B	K	S	2
03					ä	ë	Ä	Ë	c	l	t	.	C	L	T	3
04					à	è	À	È	d	m	u	ì	D	M	U	4
05					á	í	Á	Í	e	n	v	§	E	N	V	5
06						î		Î	f	o	w	ñ	F	O	W	6
07					ć	ï	Ć	Ï	g	p	x	ĵ	G	P	X	7
08					ç	ì	Ç	Ì	h	q	y	½	H	Q	Y	8
09					ñ	ß	Ñ	¨	i	r	z		I	R	Z	9
0A					`	!	^	:	Ǧ	Ş	Ɔ	Ĵ	SHY	ı	²	³
0B					.	\$	,	#	ǧ	ş	ž	[	ô	û	Ô	{
0C					<	*	%	@		ĉ		\	ö	ü	Ö	Ü
0D					(	)	_	'	ǰ	,	Ű	]	ò	Û	Ò	}
0E					+	;	>	=	š	Č	Š	'	ó	ú	Ó	Ú
0F							?	"	ñ	¤		x	ğ	.	Ğ	~



## Latin alphabet no. 4 ISO 8859-4 (CCSN: ISO88594)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00			SP	0	@	P	`	p			NBSP	°	Ā	Đ	ā	đ
01			!	1	A	Q	a	q			Ą	ą	Á	Ń	á	ń
02			"	2	B	R	b	r			κ	˘	Â	Ō	â	ō
03			#	3	C	S	c	s			Ŕ	ŕ	Ã	Ŧ	ã	ŧ
04			\$	4	D	T	d	t			ɹ	´	Ä	Ô	ä	ô
05			%	5	E	U	e	u			İ	ı	Å	Õ	å	õ
06			&	6	F	V	f	v			Ł	ł	Æ	Ö	æ	ö
07			'	7	G	W	g	w			Ş	ş	ı	×	ı	÷
08			(	8	H	X	h	x			˝	˝	Č	Ø	č	ø
09			)	9	I	Y	i	y			Š	š	É	Ÿ	é	ÿ
0A			*	:	J	Z	j	z			Ě	ě	Ę	Ú	ę	ú
0B			+	;	K	[	k	{			Ģ	ģ	Ě	Û	ë	û
0C			,	<	L	\	l				ƒ	ƒ	É	Ü	é	ü
0D			-	=	M	]	m	}			SHY	Đ	Í	Û	í	ü
0E			.	>	N	^	n	~			Ž	ž	Î	Û	î	ü
0F			/	?	O	_	o				˘	ŋ	Ī	ß	ī	˘


Extended BS2000 code **EBCDIC.DF.04-4** (CCSN: **EDF044**)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-	ø	Ø	°	ı	κ	υ	Ϛ	Ϝ	0
01					NBSP	é	/	É	a	j	ˉ	Ŕ	A	J	÷	1
02					â	ę	Â	Ę	b	k	s	Ī	B	K	S	2
03					ä	ë	Ä	Ë	c	l	t	˘	C	L	T	3
04					ā	č	Ā	Č	d	m	u	Š	D	M	U	4
05					á	í	Á	Í	e	n	v	§	E	N	V	5
06					ã	î	Ã	Î	f	o	w	ı	F	O	W	6
07					å	ı	Å	Ī	g	p	x	ł	G	P	X	7
08					ı	è	ı	É	h	q	y	Đ	H	Q	Y	8
09					ŋ	ß	Ŋ	¨	i	r	z	ž	ı	R	Z	9
0A					`	!	^	:	Ǿ	Ě	Ą	Ƒ	SHY	š	˙	ı
0B					.	\$	,	#	ǵ	ē	ŋ	[	ô	û	Ô	{
0C					<	*	%	@	đ	æ	Đ	\	ö	ü	Ö	Ü
0D					(	)	_	'	ū	,	Ū	]	ō	Ū	Ō	}
0E					+	;	>	=	ū	Æ	Ū	'	ķ	ú	Ķ	Ú
0F							?	"	ą	ą	Ž	x	ö	˙	Ō	~

Latin/Cyrillic alphabet **ISO 8859-5** (CCSN: **ISO88595**)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00			SP	0	@	P	`	p			NBSP	A	Р	a	р	№
01			!	1	A	Q	a	q			Ё	Б	С	б	с	ё
02			"	2	B	R	b	r			Ђ	В	Т	в	т	ђ
03			#	3	C	S	c	s			Ѓ	Г	У	г	у	ѓ
04			\$	4	D	T	d	t			Є	Д	Ф	д	ф	є
05			%	5	E	U	e	u			Ѕ	Е	Х	e	х	ѕ
06			&	6	F	V	f	v			І	Ж	Ц	ж	ц	і
07			'	7	G	W	g	w			Ї	З	Ч	з	ч	ї
08			(	8	H	X	h	x			Ј	И	Ш	и	ш	ј
09			)	9	I	Y	i	y			Љ	Й	Щ	й	щ	љ
0A			*	:	J	Z	j	z			Њ	К	Ъ	к	ъ	њ
0B			+	;	K	[	k	{			Ћ	Л	Ы	л	ы	ћ
0C			,	<	L	\	l				Ќ	М	Ь	м	ь	ќ
0D			-	=	M	]	m	}			ШY	Н	Э	н	э	§
0E			.	>	N	^	n	~			Ў	О	Ю	о	ю	ў
0F			/	?	O	_	o				Џ	П	Я	п	я	џ


Extended BS2000 code **EBCDIC.DF.04-5** (CCSN: **EDF045**)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-	j	и	A	E	Ъ	ль	l	й	0
01					NBSP	щ	/	Щ	a	j	џ	ѓ	A	J	ї	1
02					т	ъ	Т	Ъ	b	k	s	S	B	K	S	2
03					ф	ы	Ф	Ы	c	l	t	З	C	L	T	3
04					р	ш	Р	Ш	d	m	u	Ль	D	M	U	4
05					с	э	С	Э	e	n	v	Ї	E	N	V	5
06					у	ю	У	Ю	f	o	w	Ж	F	O	W	6
07					х	я	Х	Я	g	p	x	М	G	P	X	7
08					ч	ь	Ч	Ь	h	q	y	Н	Н	Q	Y	8
09					ё	п	б	Ј	i	r	z	О	l	R	Z	9
0A					`	!	^	:	Ѓ	Ѕ	Ё	Ќ	SHY	Й	В	Г
0B					.	\$	,	#	Л	К	П	[	е	ћ	д	{
0C					<	*	%	@	№	ц	a	\	i	ќ	ж	м
0D					(	)	_	'	§	И	н	]	ђ	л	в	}
0E					+	;	>	=	ÿ	Ц	о	Д	ѓ	њ	г	к
0F							?	"	Б	Є	Ў	з	s	џ	е	~

Extended BS2000 code **EBCDIC.EHC.LC** (CCSN: **EEHCLC**)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-	ђ	ѓ	є	s	Ђ	Ѓ	Є	S	0
01					NBSP	№	/	ё	a	j	~	Ё	A	J	§	1
02					a	и	p	ш	b	k	s	T	B	K	S	2
03					б	й	c	щ	c	l	t	У	C	L	T	3
04					в	к	т	ъ	d	m	u	Ф	D	M	U	4
05					г	л	у	ы	e	n	v	X	E	N	V	5
06					д	м	ф	ь	f	o	w	Ц	F	O	W	6
07					е	н	х	э	g	p	x	Ч	G	P	X	7
08					ж	о	ц	ю	h	q	y	Ш	H	Q	Y	8
09					з	п	ч	я	i	r	z	Щ	I	R	Z	9
0A					`	!	^	:	A	Ж	M	Ъ	SHY	ћ	J	Џ
0B					.	\$	,	#	Б	З	Н	Ы	i	ќ	Љ	[
0C					<	*	%	@	В	И	О	Ь	ï	ÿ	Њ	{
0D					(	)	_	'	Г	Й	П	Э	j	ц	Ћ	]
0E					+	;	>	=	Д	К	Р	Ю	љ	l	ќ	}
0F							?	"	Е	Л	С	Я	њ	ï	ÿ	\

Extended BS2000 code **EBCDIC.EHC.LC.1** (CCSN: **EEHCLC1**)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-	ђ	s	j	љ	ћ	Њ	Ћ	Ќ	0
01					NBSP	№	/	y	a	j	њ	ќ	A	J	Џ	1
02					a	ё	л	ђ	b	k	s	§	B	K	S	2
03					б	ж	м	ф	с	l	t	ц	C	L	T	3
04					в	з	н	х	d	m	u	Ђ	D	M	U	4
05					г	и	о	ц	e	n	v	S	E	N	V	5
06					ѓ	і	п	ч	f	o	w	J	F	O	W	6
07					д	ї	р	ш	g	p	x	Љ	G	P	X	7
08					е	й	с	щ	h	q	y	l	H	Q	Y	8
09					ё	к	т	ђ	i	r	z	ї	I	R	Z	9
0A					`	!	^	:	ы	A	E	Й	M	T	Ч	Э
0B					.	\$	,	#	ь	B	Є	[	H	У	Ш	{
0C					<	*	%	@	э	B	Ё	\	O	Ў	Щ	Ю
0D					(	)	_	'	ю	Г	Ж	]	П	Ф	Ђ	}
0E					+	;	>	=	я	Ѓ	З	К	Р	Х	Ы	Я
0F							?	"	SHY	Д	И	Л	С	Ц	Ь	~

## Latin/Greek alphabet ISO 8859-7/ELOT 928 (CCSN: ISO88579)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00			SP	0	@	P	`	p			NBSP	°	ı	Π	ü	π
01			!	1	A	Q	a	q			‘	±	Α	Ρ	α	ρ
02			"	2	B	R	b	r			’	²	Β		β	ς
03			#	3	C	S	c	s			£	³	Γ	Σ	γ	σ
04			\$	4	D	T	d	t			€	´	Δ	Τ	δ	τ
05			%	5	E	U	e	u			ƒ	ˆ	Ε	Υ	ε	υ
06			&	6	F	V	f	v			ı	À	Z	Φ	ζ	φ
07			'	7	G	W	g	w			§	·	H	X	η	χ
08			(	8	H	X	h	x			¨	È	Θ	Ψ	θ	ψ
09			)	9	I	Y	i	y			©	Ή	Ι	Ω	ι	ω
0A			*	:	J	Z	j	z			„	Ì	K	Ϊ	κ	ϊ
0B			+	;	K	[	k	{			«	»	Λ	ÿ	λ	ü
0C			,	<	L	\	l				¬	Ò	M	ά	μ	ό
0D			-	=	M	]	m	}			SHY	½	N	έ	ν	ύ
0E			.	>	N	^	n	~				Υ	Ξ	ή	ξ	ώ
0F			/	?	O	_	o				–	Ω	Ο	ί	ο	

Extended BS2000 code **EBCDIC.DF.04-7** (CCSN: **EDF047**)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-	ψ	Ψ	°	ˆ	'	ω	ı	Ω	0
01					NBSP	ı	/	ı	a	j	-	£	A	J	χ	1
02					β	κ	B	K	b	k	s	Đp	B	K	S	2
03					δ	λ	Δ	Λ	c	ı	t	·	C	L	T	3
04					ü	θ	ı	Θ	d	m	u	©	D	M	U	4
05					α	v	A	N	e	n	v	§	E	N	V	5
06					γ	ξ	Γ	Ξ	f	o	w	À	F	O	W	6
07					ε	o	E	O	g	p	x	Ó	G	P	X	7
08					η	μ	H	M	h	q	y	½	H	Q	Y	8
09					ρ	ı	P	ˆ	ı	r	z	Ÿ	ı	R	Z	9
0A					`	!	^	:	«	ˆ	'	¬	SHY	Ĥ	²	³
0B					.	\$	,	#	»	ı	Ω	[	τ	ü	T	{
0C					<	*	%	@	π	ζ	Π	\	φ	ó	Φ	á
0D					(	)	_	'	ú	Ě	é	]	ς	ÿ		}
0E					+	;	>	=	ώ	Z	ή	'	σ	ı	Σ	İ
0F							?	"	±	€		X	u		Y	~



Extended BS2000 code **EBCDIC.EHC.LG** (CCSN: **EEHCLG**)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-	NBSP	–	°		±		¡		0
01					A	H	/		a	j		½	A	J	©	1
02					À	Θ	Ξ	Υ	b	k	s	²	B	K	S	2
03					B	I	O	Υ	c	l	t	³	C	L	T	3
04					Γ	Ι	Ο	ÿ	d	m	u	´	D	M	U	4
05					Δ	Ï	Π	Φ	e	n	v	´	E	N	V	5
06					E	K	P	X	f	o	w	§	F	O	W	6
07					È	Λ		Ψ	g	p	x	·	G	P	X	7
08					Z	M	Σ	Ω	h	q	y	£	H	Q	Y	8
09					H	N	T	Ω	i	r	z	´	I	R	Z	9
0A					`	!	^	:	α	é	í	¬	v	ς	ü	·
0B					.	\$	,	#	á	ζ	ï	[	ξ	σ	φ	{
0C					<	*	%	@	β	η	ï	\	o	τ	χ	¨
0D					(	)	_	'	γ	ή	κ	]	ó	u	ψ	}
0E					+	;	>	=	δ	θ	λ	«	π	ú	ω	¨
0F						SHY	?	"	ε	ι	μ	»	ρ	ü	ώ	~

## 7.3.1.2 7-bit line codes and associated EBCDIC codes for European alphabets

## Swedish 7-bit code

ISO 646, Swedish version for names / SEN 850200

	00	10	20	30	40	50	60	70
00			SP	0	É	P	é	p
01			!	1	A	Q	a	q
02			"	2	B	R	b	r
03			#	3	C	S	c	s
04			¤	4	D	T	d	t
05			%	5	E	U	e	u
06			&	6	F	V	f	v
07			'	7	G	W	g	w
08			(	8	H	X	h	x
09			)	9	I	Y	i	y
0A			*	:	J	Z	j	z
0B			+	;	K	Ä	k	ä
0C			,	<	L	Ö	l	ö
0D			-	=	M	Å	m	å
0E			.	>	N	Ü	n	ü
0F			/	?	O	_	o	

BS2000 code **EBCDIC.DF.03.SWE**

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-									0
01							/		a	j			A	J		1
02									b	k	s		B	K	S	2
03									c	l	t		C	L	T	3
04									d	m	u		D	M	U	4
05									e	n	v		E	N	V	5
06									f	o	w		F	O	W	6
07									g	p	x		G	P	X	7
08									h	q	y		H	Q	Y	8
09									i	r	z		I	R	Z	9
0A					é	!	Ü	:								
0B					.	\$	,	#				Ä				ä
0C					<	*	%	É				Ö				
0D					(	)	_	'				Å				å
0E					+	;	>	=								
0F					ö		?	"								ü

BS2000 code **EBCDIC.NHC.SWE**

Characters comply with ISO 646-SWE

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-						ä	å	É	0
01							/		a	j	ü		A	J		1
02									b	k	s		B	K	S	2
03									c	l	t		C	L	T	3
04									d	m	u		D	M	U	4
05									e	n	v		E	N	V	5
06									f	o	w		F	O	W	6
07									g	p	x		G	P	X	7
08									h	q	y		H	Q	Y	8
09							é		i	r	z		I	R	Z	9
0A					#	α	ö	:								
0B					.	Å	,	Ä								
0C					<	*	%	Ö								
0D					(	)	_	'								
0E					+	;	>	=								
0F					!	Ü	?	"								

Hungarian 7-bit code Siemens version **TD7.HUN**

	00	10	20	30	40	50	60	70
00			SP	0	@	P	`	
01			!	1	A	Q	Ft	
02			"	2	B	R	Á	
03			#	3	C	S	É	
04			¤	4	D	T	Í	
05			%	5	E	U	Ó	
06			&	6	F	V	Ö	
07			'	7	G	W	Ő	
08			(	8	H	X	Ú	
09			)	9	I	Y	Ü	
0A			*	:	J	Z	Û	
0B			+	;	K	[		{
0C			,	<	L	\		
0D			-	=	M	]		}
0E			.	>	N	^		~
0F			/	?	O	_		

BS2000 code **EBCDIC.DF.03.HUN**

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-									0
01							/		Ft	Ű			A	J		1
02									Á				B	K	S	2
03									É				C	L	T	3
04									í				D	M	U	4
05									Ó				E	N	V	5
06									Ö				F	O	W	6
07									Ő				G	P	X	7
08									Ú				H	Q	Y	8
09									Ü				I	R	Z	9
0A					`	!	^	:								
0B					.	¤	,	#				[				{
0C					<	*	%	@				\				
0D					(	)	_	'				]				}
0E					+	;	>	=								
0F							?	"								~

BS2000 code **EBCDIC.NHC.HUN**


	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-									0
01							/					Á	A	J		1
02												É	B	K	S	2
03												Í	C	L	T	3
04												Ó	D	M	U	4
05												Ö	E	N	V	5
06												Ő	F	O	W	6
07												Ú	G	P	X	7
08												Ü	H	Q	Y	8
09					Ft							Ű	I	R	Z	9
0A					`	!	^	:								
0B					.	¤	,	#				[				{
0C					<	*	%	@				\				
0D					(	)	_	'				]				}
0E					+	;	>	=								
0F							?	"								~

Cyrillic 7-bit code Siemens version **TD7.CYR**

	00	10	20	30	40	50	60	70
00			SP	0	@	P	Ю	П
01			!	1	A	Q	А	Я
02			"	2	B	R	Б	Р
03			#	3	C	S	Ц	С
04			¤	4	D	T	Д	Т
05			%	5	E	U	Е	У
06			&	6	F	V	Ф	Ж
07			'	7	G	W	Г	В
08			(	8	H	X	Х	Ш
09			)	9	I	Y	И	З
0A			*	:	J	Z	Й	Щ
0B			+	;	K	Ы	К	[
0C			,	<	L	Ь	Л	
0D			-	=	M	Э	М	]
0E			.	>	N	^	Н	Ч
0F			/	?	O	_	O	



BS2000 code **EBCDIC.DF.03.CYR**

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-									0
01							/		А	Й			А	Ј		1
02									Б	К	С		В	К	Ѕ	2
03									Ц	Л	Т		С	Л	Т	3
04									Д	М	У		Д	М	У	4
05									Е	Н	Ж		Е	Н	Ѳ	5
06									Ф	О	В		Ф	О	Ѳ	6
07									Г	П	Ш		Г	Р	Х	7
08									Х	Я	З		Н	Q	Y	8
09									И	Р	Ц		І	Р	Z	9
0A					Ю	!	^	:								
0B					.	¤	,	#				Ы				[
0C					<	*	%	@				Ь				
0D					(	)	_	'				Э				]
0E					+	;	>	=								
0F							?	"								Ч


BS2000 code **EBCDIC.NHC.CYR**

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-									0
01							/						A	J		1
02												Т	B	K	S	2
03												У	C	L	T	3
04												Ф	D	M	U	4
05												Х	E	N	V	5
06												Ц	F	O	W	6
07												Ч	G	P	X	7
08												Ш	H	Q	Y	8
09												Щ	I	R	Z	9
0A						!	^	:	А	Ж	М					
0B					.	¤	,	#	Б	З	Н	Ы				[
0C					<	*	%	@	В	И	О	Ь				
0D					(	)	_	'	Г	Й	П	Э				]
0E					+	;	>	=	Д	К	Р	Ю				
0F							?	"	Е	Л	С	Я				

Greek 7-bit code Siemens version **TD7.GRE**

	00	10	20	30	40	50	60	70
00			SP	0	@	P	`	Π
01			!	1	A	Q	A	P
02			"	2	B	R	B	C
03			#	3	C	S	Γ	Σ
04			\$	4	D	T	Δ	T
05			%	5	E	U	E	Υ
06			&	6	F	V	Z	Φ
07			'	7	G	W	H	X
08			(	8	H	X	Θ	Ψ
09			)	9	I	Y	I	Ω
0A			*	:	J	Z	K	§
0B			+	;	K	[	Λ	{
0C			,	<	L	\	M	
0D			-	=	M	]	N	}
0E			.	>	N	^	≡	~
0F			/	?	O	_	O	

BS2000 code **EBCDIC.DF.03.GRE**

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-									0
01							/		A	K			A	J		1
02									B	Λ	Σ		B	K	S	2
03									Γ	M	T		C	L	T	3
04									Δ	N	Υ		D	M	U	4
05									E	Ξ	Φ		E	N	V	5
06									Z	O	X		F	O	W	6
07									H	Π	Ψ		G	P	X	7
08									Θ	P	Ω		H	Q	Y	8
09									I		§		I	R	Z	9
0A					`	!	^	:								
0B					.	\$	,	#				[				{
0C					<	*	%	@				\				
0D					(	)	_	'				]				}
0E					+	;	>	=								
0F							?	"								~

BS2000 code **EBCDIC.NHC.GRE**

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-									0
01					A		/						A	J		1
02						Θ	Ξ	Υ					B	K	S	2
03					B	I	O						C	L	T	3
04					Γ								D	M	U	4
05					Δ		Π	Φ					E	N	V	5
06					E	K	P	X				§	F	O	W	6
07						Λ		Ψ					G	P	X	7
08					Z	M	Σ	Ω					H	Q	Y	8
09					H	N	T						I	R	Z	9
0A					`	!	^	:								
0B					.	\$	,	#				[				{
0C					<	*	%	@				\				
0D					(	)	_	'				]				}
0E					+	;	>	=								
0F							?	"								~

International 7-bit code **ISO 646-IRV** (CCSN: **ISO646**)

	00	10	20	30	40	50	60	70
00			SP	0	@	P	`	p
01			!	1	A	Q	a	q
02			"	2	B	R	b	r
03			#	3	C	S	c	s
04			¤	4	D	T	d	t
05			%	5	E	U	e	u
06			&	6	F	V	f	v
07			'	7	G	W	g	w
08			(	8	H	X	h	x
09			)	9	I	Y	i	y
0A			*	:	J	Z	j	z
0B			+	;	K	[	k	{
0C			,	<	L	\	l	
0D			-	=	M	]	m	}
0E			.	>	N	^	n	~
0F			/	?	O	_	o	

BS2000 code **EBCDIC.DF.03** international

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-									0
01							/		a	j			A	J		1
02									b	k	s		B	K	S	2
03									c	l	t		C	L	T	3
04									d	m	u		D	M	U	4
05									e	n	v		E	N	V	5
06									f	o	w		F	O	W	6
07									g	p	x		G	P	X	7
08									h	q	y		H	Q	Y	8
09									i	r	z		I	R	Z	9
0A					`	!	^	:								
0B					.	\$	,	#				[				{
0C					<	*	%	@				\				
0D					(	)	_	'				]				}
0E					+	;	>	=								
0F							?	"								~

### 7.3.2 Supported line codes and BS2000 EBCDIC codes for Arabic alphabets

Key:



Control character cannot be assigned



Position cannot be assigned



EBCDIC.DF03 kernel



This special character is not supported in the Arabic script



Selectable display in Arabic script



Compatibility-relevant Siemens extension



Latin character used in Arabic script



Latin/Arabic alphabet ISO 8859-6 / TD8.LA (CCSN: ISO88596)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00			SP	0 .	@	P	`	p						ذ	-	ـ
01			!	1 ١	A	Q	a	q					ء	ر	ف	ء
02			"	2 ٢	B	R	b	r					آ	ز	ق	٥
03			#	3 ٣	C	S	c	s					أ	س	ك	لا
04			\$	4 ٤	D	T	d	t			ء		و	ش	ل	لا
05			%	5 ٥	E	U	e	u					إ	ص	م	لا
06			&	6 ٦	F	V	f	v					ئ	ض	ن	لا
07			'	7 ٧	G	W	g	w					ا	ط	ه	
08			(	8 ٨	H	X	h	x					ب	ظ	و	
09			)	9 ٩	I	Y	i	y					ة	ع	ى	
0A			*	:	J	Z	j	z					ت	غ	ي	
0B			+	;	K	[	k	{				؛	ث		"	
0C			,	<	L	\	l				،		ج		"	
0D			-	=	M	]	m	}					ح		"	
0E			.	>	N	^	n	~					خ		'	
0F			/	?	O	_	o					؟	د		ء	

Extended BS2000 code **EBCDIC.DF.04-6** (CCSN: **EDF046/EDF04A**)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0			
00					SP	&	-	ظ	0	.	5	0	"	&	ع	0			
01					SP	ى	/	ة	a	j	/	#	A	J		1			
02					ق	ي	آ	ت	b	k	s	%	B	K	S	2			
03					ل	ء	ؤ	ث	c	l	t	7	v	C	L	T	3		
04					-	و	@	ب	d	m	u	)	D	M	U	4			
05					ف	ء	ء	ح	e	n	v	'	E	N	V	5			
06					ك	'	أ	خ	f	o	w	6	٦	F	O	W	6		
07					م	ء	إ	د	g	p	x	<	G	P	X	7			
08					ه	ء	ا	ج	h	q	y	=	H	Q	Y	8			
09					ء	-	ر	(	i	r	z	>	I	R	Z	9			
0A					`	!	^	:	+	*	!	,	-	9	٩	2	٢	3	٣
0B					.	\$	,	#	:	:	?	[	لأ	{	ش	{			
0C					<	*	%	@	,	ن	ذ	\	لا		ض	\			
0D					(	)	-	'	}	8	٨	]	]	°	[	ز	}		
0E					+	;	>	=	-	ى	^	4	٤	لآ		س	غ		
0F							?	"	1	١	٥	.	ط	لا		ص	~		

Extended BS2000 code **EBCDIC.EHC.LA** (CCSN: **EEHCLAA/EEHCLAI**)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0	
00					SP	&	-	ف				0	.	-	&	SP	0
01					ء	ت	/	ق	a	j		1	١	A	J	/	1
02					آ	ث	س	ك	b	k	s	2	٢	B	K	S	2
03					أ	ح	ش	ل	c	l	t	3	٣	C	L	T	3
04					ؤ	ح	ص	م	d	m	u	4	٤	D	M	U	4
05					إ	خ	ض	ن	e	n	v	5	٥	E	N	V	5
06					ئ	د	ط	ه	f	o	w	6	٦	F	O	W	6
07					ا	ذ	ظ	و	g	p	x	7	٧	G	P	X	7
08					ب	ر	ع	ى	h	q	y	8	٨	H	Q	Y	8
09					ة	ز	غ	ي	i	r	z	9	٩	I	R	Z	9
0A					،	!	^	:	}	"	'	.	:	^	!		
0B					.	\$	,	#	]	"	'	[	#	,	¤	{	
0C					<	*	%	@	\	"	'	\	@	%	*	>	
0D					(	)	-	'	[	"	'	]	'	-	(	}	
0E					+	;	>	=	{	"	'	}	=	<	:	+	
0F						-	?	"		'	"	-	"	?		~	

## French/Arabic alphabet TD8.NA

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00			SP	0 .	@	P	`	p						ذ	-	ـ
01			!	1 ١	A	Q	a	q					ء	ر	ف	ـ
02			"	2 ٢	B	R	b	r					آ	ز	ق	°
03			#	3 ٣	C	S	c	s					أ	س	ك	Ç
04			\$	4 ε	D	T	d	t			α		ؤ	ش	ل	à
05			%	5 ٥	E	U	e	u					إ	ص	م	â
06			&	6 ٦	F	V	f	v					ى	ض	ن	ç
07			'	7 ٧	G	W	g	w					ا	ط	ه	è
08			(	8 ٨	H	X	h	x					ب	ظ	و	é
09			)	9 ٩	I	Y	i	y					ة	ع	ى	ê
0A			*	:	J	Z	j	z					ت	غ	ي	ë
0B			+	;	K	[	k	{				:	ث	ـ	ـ	î
0C			,	<	L	\	l						ج	°	ـ	ï
0D			-	=	M	]	m	}					ح	ـ	ـ	ô
0E			.	>	N	^	n	~					خ	§	ـ	ù
0F			/	?	O	_	o					?	د	ـ	ـ	û

Extended BS2000 code **EBCDIC.DF.04-NAF** (CCSN: EDF04E/EDF04F)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0			
00					SP	&	-	é	ط	0	.	5	"	ê	&	ε	0		
01					SP	ى	/	ة	a	j	/	#	A	J	è		1		
02					ق	ي	آ	ت	b	k	s	%	B	K	S		2		
03					ج	هـ	ؤ	ث	c	l	t	7	v	C	L	T	3		
04					-	و	@	ب	d	m	u	)	D	M	U		4		
05					ف	هـ	°	ح	e	n	v	'	E	N	V		5		
06					ك	ـ	أ	خ	f	o	w	6	٦	F	O	W	6		
07					م	س	إ	د	g	p	x	<	G	P	X		7		
08					هـ	ع	ا	ج	h	q	y	=	H	Q	Y		8		
09					ع	-	ر	(	i	r	z	>	I	R	Z		9		
0A					,	!	^	:	+	*	!	,	-	9	٩	2	٢	3	٣
0B					.	\$	'	#	:	:	?	[	à	î	ش	{			
0C					<	*	%	@	,	ن	ذ	\	ç	ï	ض	هـ			
0D					(	)	-	'	ô	8	٨	¶	]	°	¨	ز	}		
0E					+	;	>	=	ù	ى	§	4	ε	Ç	ë	س	غ		
0F							?	"	1	١	α	.	ط	â	û	ص	~		

Extended BS2000 code **EBCDIC.EHC.NA** (CCSN: **EEHCNAA/EEHCNAI**)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-	ف	§	▨	ë	0 .	-	&	SP	0
01					°	ت	/	ق	a	j	î	1 ı	A	J	/	1
02					آ	ث	س	ك	b	k	s	2 ۲	B	K	S	2
03					أ	ج	ش	ل	c	l	t	3 ۳	C	L	T	3
04					ؤ	ح	ص	م	d	m	u	4 ε	D	M	U	4
05					إ	خ	ض	ن	e	n	v	5 ۵	E	N	V	5
06					ى	د	ط	ه	f	o	w	6 ٦	F	O	W	6
07					ا	ذ	ظ	و	g	p	x	7 ۷	G	P	X	7
08					ب	ر	ع	ى	h	q	y	8 ٨	H	Q	Y	8
09					ة	ز	غ	ي	i	r	z	9 ٩	I	R	Z	9
0A					`	!	^	:	à	ˆ	˘	.	:	Ç	!	¨
0B					.	\$	،	#	â	ˆ	˚	[	#	،	¤	{
0C					<	*	%	@	ç	ˆ	ï	\	@	%	*	>
0D					(	)	_	'	è	ˆ	ô	]	'	_	(	}
0E					+	;	>	=	é	ˆ	ù	)	=	<	:	+
0F						-	?	"	ê	ˆ	û	¶	"	?	°	~

### 7.3.3 Supported line codes and BS2000 EBCDIC codes for Persian alphabets

Key:



Control character cannot be assigned



Position cannot be assigned



EBCDIC.DF03 kernel



This special character is not supported in Farsi script



Selectable display in Farsi script



Required for Farsi compounds



Latin character used in Farsi script

## Latin/Farsi alphabet for data interchange TD8.LF

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00			SP	0 .	@	P	`	p					ء	چ	ض	گ
01			!	1 ۱	A	Q	a	q					آ	ح	ض	گ
02			"	2 ۲	B	R	b	r					ا	ح	ط	ل
03			#	3 ۳	C	S	c	s					ا	خ	ط	ل
04			\$	4 ۴	D	T	d	t			رول		آ	خ	ظ	م
05			%	5 ۵	E	U	e	u					ب	د	ظ	م
06			&	6 ۶	F	V	f	v			÷		ب	ذ	ع	ن
07			'	7 ۷	G	W	g	w					پ	ر	ع	ن
08			(	8 ۸	H	X	h	x					پ	ز	غ	و
09			)	9 ۹	I	Y	i	y					ت	ژ	غ	ؤ
0A			*	:	J	Z	j	z			x		ت	س	ف	ه
0B			+	;	K	[	k	{				؛	ث	س	ف	ه
0C			,	<	L	\	l				،		ث	ش	ق	ی
0D			-	=	M	]	m	}					ج	ش	ق	ی
0E			.	>	N	^	n	~					ج	ص	ک	ئ
0F			/	?	O	_	o					؟	چ	ص	ک	_



Extended BS2000 code **EBCDIC.DF.04-FAR** (CCSN: EDF04C/EDF04D)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0			
00					SP	&	-	و	ز	0	.	5	و	ؤ	÷	ژ	0		
01					SP	غ	/	ت	a	j	/	#	A	J	ن		1		
02					ط	ف	ا	ت	b	k	s	%	B	K	S		2		
03					ظ	ف	أ	ث	c	l	t	7	v	C	L	T	3		
04					ض	غ	ء	پ	d	m	u	)	D	M	U		4		
05					ض	ق	آ	ج	e	n	v	'	E	N	V		5		
06					ط	ک	ا	ج	f	o	w	6	٦	F	O	W	6		
07					ظ	ک	ب	ج	g	p	x	<	G	P	X		7		
08					ع	ق	پ	ث	h	q	y	=	H	Q	Y		8		
09					گ	ص	ح	(	i	r	z	>	I	R	Z		9		
0A					`	!	^	:	+	x	!	,	-	9	٩	2	٢	3	٣
0B					.	\$	,	#	:	:	?	[	م	ه	خ	{			
0C					<	*	%	@	گ	ع	چ	\	ن	ی	ذ	ش			
0D					(	)	-	'	ی	8	٨	ش	]	ل	س	ح	}		
0E					+	;	>	=	ئ	ب	ص	4	٤	ل	ه	خ	س		
0F							?	"	1	١	روال	.	ر	م	-	د	~		

## Extended BS2000 code EBCDIC.EHC.LF (CCSN: EEHCLFI/EEHCLFF)

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
00					SP	&	-	6 ٦	ب		خ	س	غ	ك	SP	0
01					#	¤	/	7 ٧	a	j	خ	ش	A	J	ن	1
02					'	%	/	8 ٨	b	k	s	ش	B	K	S	2
03					÷	.	?	9 ٩	c	l	t	ص	C	L	T	3
04					<	:	0 ٠	ء	d	m	u	ص	D	M	U	4
05					:	)	1 ١	آ	e	n	v	ض	E	N	V	5
06					>	(	2 ٢	ا	f	o	w	ض	F	O	W	6
07					!	x	3 ٣	ا	g	p	x	ط	G	P	X	7
08					"	+	4 ٤	أ	h	q	y	ط	H	Q	Y	8
09					,	-	5 ٥	ب	i	r	z	ظ	I	R	Z	9
0A					`	!	^	:	پ	ج	د	ظ	غ	گ	ن	ی
0B					.	\$	,	#	پ	ج	ذ	[	ف	گ	و	{
0C					<	*	%	@	ت	چ	ر	\	ف	ل	ؤ	ئ
0D					(	)	_	'	ت	چ	ز	]	ق	ل	ه	}
0E					+	;	>	=	ث	ح	ژ	ع	ق	م	ه	-
0F						=	?	"	ث	ح	س	ع	ك	م	ی	~

---

# Glossary

## **ASCII**

U.S. version of the 7-bit code table based on ISO646.

## **character set**

Set of letters, digits and special characters used to create words and other components of a language (or computer language).

## **code**

See "Coded character set".

## **code tables**

Tables that describe a code. A number of tables are assigned to each code described. These tables are created and modified using macros.

## **coded character set**

A set of rules that assigns the characters of a character set uniquely and specifies their representation in bits.

## **communication access method**

Software that offers applications interfaces for communication.

## **compatible coded character sets**

Codes are compatible with each other if they have the same character sets. However, the coding or values of the individual characters can differ from one set to the other.

## **defined character**

A particular value of a code is said to be defined if it has been assigned a (printable or nonprintable) character.

## **displayable character**

Character that can be represented graphically (including blank).

## **EBCDIC kernel**

Standardized basic BS2000 character set comprising the following characters: A-Z a-z 0-9 blank & - / : . , < \* % ( ) \_ ' + ; > = ?

**extension**

A coded character set B is an extension of the coded character set A if it contains all the characters defined in A plus some additional characters. Characters common to both codes must be coded in the same way.

**format terminal**

Operating mode of a virtual terminal in which the message consists of a format (a form or mask).

**ISO code**

Standard code for communication used, for example, in UNIX, MS-DOS and DEC systems. ISO is derived from ASCII.

**ISO code variant number**

Number that identifies a standardized variant of the ISO8859 code and the corresponding compatible EBCDIC code.

**line terminal**

Operating mode of a virtual terminal, in which the message is structured in the form of lines.

**message**

A logically related volume of data sent to or received from a communication partner.

**printable character**

See "displayable character".

**process**

Module required to execute a program within a task.

**protocol**

A set of rules regulating communication in computer networks.

**reference code**

In a family of compatible codes, the conversion tables refer to a common target code: the family's reference code. The reference code contains all the characters of the other codes.

**special character**

A displayable character that is neither a letter, a digit or a blank.

**status query**

A status query provides you with basic information on the data display terminal (e.g. terminal type, the number of logically accessible character sets with their types, and the number of colors) and a description of the peripherals connected (e.g. ID card readers, chipcard terminals).

**virtual terminal**

Idealized terminal whose functions are mapped to the physical attributes of different terminals.

**7-bit code**

An ISO/ASCII code that uses only 7 bits, or the equivalent EBCDIC code.

**8-bit code**

An ISO code that uses 8 bits, or the equivalent EBCDIC code.



---

# Abbreviations

<b>BCAM</b>	Basic Communication Access Method
<b>CCS</b>	Coded Character Set
<b>CCSN</b>	Coded Character Set Name
<b>DCAM</b>	Data Communication Access Method
<b>DRV</b>	German reference version
<b>DSSM</b>	Dynamic SubSystem Management
<b>EBCDIC</b>	Extended Binary-Coded Decimal Interchange Code
<b>EMDS</b>	Data display terminal emulation (German acronym)
<b>IRV</b>	International Reference Version
<b>ISO</b>	international organization for standardization
<b>NLS</b>	Native Language Support
<b>PDN</b>	program system for teleprocessing and network control (German acronym)
<b>TIAM</b>	Terminal Interactive Access Method
<b>openUTM</b>	Universal Transaction Monitor
<b>UCS</b>	Universal Character Set
<b>UTF-8</b>	Universal Character Set Transformation Format 8 Bit
<b>UTF-16</b>	Universal Character Set Transformation Format 16 Bit
<b>UTFE</b>	Universal Character Set Transformation Format EBCDIC

## Abbreviations

---

**VTSU** Virtual Terminal Support

**VTSUCB** VTSU Control Block



---

## Related publications

### Ordering manuals

The manuals are available as online manuals, see <http://manuals.fujitsu-siemens.com>, or in printed form which must be paid and ordered separately at <http://FSC-manualshop.com>.

- [1] **VTSU**  
Virtual Terminal Support  
User Guide
- [2] **BS2000/OSD-BC**  
Introductory Guide to Systems Support  
User Guide
- [3] **BS2000/OSD-BC**  
System Installation  
User Guide
- [4] **BS2000/OSD-BC**  
**Commands**  
Volumes 1 - 5  
User Guide
- [5] **BS2000/OSD-BC V6.0**  
**Commands, Volume 6, Output in S Variables and SDF-P-BASYS**  
User Guide
- [6] **BS2000/OSD-BC**  
Utility Routines  
User Guide
- [7] **BS2000/OSD-BC**  
**Executive Macros**  
User Guide

- [8] **BS2000/OSD-BC**  
System Exits  
User Guide
  
- [9] **IMON (BS2000/OSD)**  
Installation Monitor  
User Guide
  
- [10] **EMDS (SINIX)**  
User Guide
  
- [11] *openNet Server* (BS2000/OSD)  
**BCAM**  
User Guide
  
- [12] **Generating a Data Communication System (TRANSDATA)**  
User Guide
  
- [13] **TransView-NMA/NMAE V1.2A, TransView-NTAC2 V7.1A, NTAC2E V5.1A**  
(TRANSDATA, BS2000)  
Network Management in BS2000  
User Guide
  
- [14] **TransView NMA/NMAE (PDN)**  
Network Management and Measurement Data Compilation in PDN (TRANSDATA, PDN)  
Commands  
User Guide
  
- [15] **Network Access for Terminals (TRANSDATA)**  
User's Guide
  
- [16] **SDF (BS2000/OSD)**  
SDF Management
  
- [17] **DCAM (BS2000/OSD, TRANSDATA)**  
Macros  
User Guide
  
- [18] **DCAM (BS2000/OSD, TRANSDATA)**  
COBOL Calls  
User Guide
  
- [19] **DCAM (BS2000/OSD, TRANSDATA)**  
Program Interfaces  
Reference Manual

- [20] **TIAM** (TRANSDATA, BS2000/OSD)  
User Guide
- [21] **FHS** (TRANSDATA)  
User Guide
- [22] **IFG for FHS** (TRANSDATA)  
User Guide
- [23] **openUTM** (BS2000/OSD, UNIX, Windows)  
Generating Applications  
User Guide
- [24] **openUTM** (BS2000/OSD, UNIX, Windows)  
Programming Applications with KDCS for COBOL, C and C++  
User's Guide
- [25] **openUTM**  
Concepts and Functions  
User Guide
- [26] **EDT** (BS2000/OSD)  
Statements  
User Guide
- [27] **PERCON** (BS2000/OSD)
- [28] **SORT** (BS2000/OSD)  
User Guide
- [29] **RSO** (BS2000/OSD)  
Remote SPOOL Output  
User Guide
- [30] **LMS** (BS2000/OSD)  
SDF Format  
User Guide
- [31] **BS2000/OSD**  
**Softbooks English**  
CD-ROM
- [32] **Unicode in BS2000/OSD**  
Introduction

## Literature on Unicode on the Internet

<http://www.unicode.org>

Home page of the Unicode consortium

<http://www.unicode.org/reports/tr10/>

Unicode Collation Algorithm

<http://www.unicode.org/reports/tr15/>

Unicode Normalization Forms

<http://www.unicode.org/reports/tr16/>

UTF-EBCDIC

<http://www.unicode.org/charts>

Unicode Code Charts

<http://www.unicode.org/Public/UCA/4.0.0/allkeys-4.0.0.txt>

Unicode Default Collation Table

<http://www.unicode.org/Public/UNIDATA/UnicodeData.txt>

Unicode Character Database

<http://developer.mimer.com/collations/charts/index.tml>

Amongst others specific national sorting rules

<http://unicode.e-workers.de/>

General introduction in Unicode

---

# Index

7-/8-bit character sets  
    unambiguously reversible conversion 107  
7-bit codes 143  
7-bit mode 31, 55, 56  
7-bit printers 32  
7-bit terminal 31  
8-bit code  
    defining 130  
8-bit codes 16, 144  
    Arabic 36  
    Persian 36  
8-bit mode 30, 31, 55, 56  
    activating 45  
    automatic activation 46  
    explicit activation 45  
    implicit activation 46  
    permanent 46, 140  
    program-related 140  
8-bit printers 32  
8-bit terminal 25, 30, 31, 140  
    Arabic 36  
    Persian 36, 141

## A

add dynamically  
    new coded character sets 132  
ADD-CODE-TABLES 132  
Arabic 8-bit code 36  
Arabic 8-bit data display terminal 36  
Arabic codes 134  
    incompatibilities 138  
ASCII code 16

## B

blank code name 34

branching from TU without SVC (NLSCNV) 105  
BS2000 command  
    ADD-CODE-TABLES 132

## C

CCS 15  
CCSN 15, 45, 144  
    determining 54  
    empty 55  
    of output files 55, 56  
    of statements 55, 56  
centralizing code tables 46  
character set  
    coded 15  
    extended 16  
character set name 15  
character value 116, 129  
code compatibility 79  
code conversion 39  
code extension mechanism 39  
code group 115  
code information 64  
code name  
    blank 34  
code names 113  
code tables 28, 113  
    centralizing 46  
code transparency 47  
coded character set 15  
    adding dynamically 132  
codes  
    fully compatible 115  
    partly compatible 115  
collation element 131  
collation entries 131

compatibility check 30, 48, 50, 52, 53  
complete code name 144  
complete codes 55  
concept alphabets 17  
configuration file 35  
connection characteristics 33  
conversion  
    unambiguously reversible (7-/8-bit character sets) 107  
conversion table 115, 116  
converting  
    lowercase letters to uppercase 117  
    maximum length of the output string 102  
    to compatible target code 101  
    to Unicode 118  
converting lowercase to uppercase 103  
creation  
    of code tables 119  
currency symbol 137

## D

data code 15  
data display terminals 25  
DCAM 30  
DCAM application 50, 53  
DCSTA macro 33, 44  
defining  
    8-bit code 130  
    Unicode characters 130  
detecting ISO codes 54  
determining the name of the data character set 54  
DSSM 28  
dynamic subsystem 28

## E

EBCDIC 16  
EBCDIC.DF.04 28, 39, 115  
EBCDIC.DF.04-1 16  
EDF03IRV 28  
EDT 57  
empty CCSN 55  
ESC printers 142

Escape printers 35  
ESCAPE sequences 34  
euro symbol 136  
    emulation 136  
European 7-bit terminal 35, 141  
extended character sets 15, 16  
extended terminal protocol 44

## F

FHS 48, 58  
FHS standard table set 58  
file attribute 45  
FT-BS2000 60  
fully compatible codes 115

## G

generating sets of tables 120  
GNLADPT 106  
GNLMTAB 28, 113, 121, 140  
    user-defined 128  
group reference code 115  
grouping compatible codes 16, 115

## I

ICE 57  
ideographic alphabets 17  
IFG 48, 57  
information about codes 55, 56  
input and output strings  
    length 108  
input string  
    normalizing 103  
installation 27  
installation procedure 35, 141  
interfaces (BS2000/OSD)  
    Unicode encodings 43  
ISO 8859 16, 39  
ISO code 144  
ISOVAR 83

## K

KDCFILE 51

**L**

length of the input and output strings  
  length 108  
letter alphabets 17  
library  
  user-specific 128  
library elements 45  
link characteristics 44  
LMS 59  
lowercase letters to uppercase letters  
  converting 103

**M**

message header 52, 53  
mode switch 34  
modification  
  of code tables 119  
module library  
  default 128  
  user-specific 128  
multibyte code processing 17

**N**

national 7-bit support 35  
NLSCCS 121, 122  
NLSCMP 79  
NLSCNV 87  
  branching from TU without SVC 105  
NLSCOD 64  
NLSCTAB 119, 120  
NLSHEAD 121  
nonprivileged programs 28  
normalization 19, 103  
  maximum length of the output string 104  
notational conventions 12

**O**

OMNIS 60  
openUTM 30  
openUTM application 51, 53  
output strings  
  length 108

**P**

partly compatible codes 115  
PDN free text parameter 35, 139, 141  
PDN generation 39, 139, 141  
PERCON 58  
permanent 8-bit mode 46, 140  
Persian 8-bit code 36  
Persian 8-bit terminal 36, 141  
Persian codes 134  
PLAM library elements 45, 54  
printers 25  
privileged programs 28  
programming notes 34  
program-related 8-bit mode 140  
programs  
  nonprivileged 28  
  privileged 28

**R**

reference code 115  
  conversion 118  
restricted codes 55  
return information 146  
RSO 59

**S**

SHOW-FILE command 57  
single-byte code processing 17  
SORT 59  
sort information  
  defining (Unicode) 131  
sort sequence 59  
sort table 117  
standard CCSN 143  
status query 30, 48, 50, 52, 53  
string conversion 87  
subsystem, dynamic 28  
supervisor call 83  
support  
  Unicode 18  
SVC 83  
SYSFILE GCCSN 54  
system default code 113

## T

TABLE 74

table

- converting a code to the reference code of the family 116
- converting lowercase letters to uppercase 116
- converting lowercase to uppercase 117
- converting the reference code 116
- converting to reference code 116
- converting to Unicode 116
- of character properties 117
- properties 116
- structure 116

terminal

- 7-bit European 141
- 8-bit capable 25
- characteristics 33, 44

terminal characteristics 33, 44

terminal emulation MT9750 37

terminal protocol

- extended 44

terminal status 33, 44

TIAM 30

TIAM application 48, 52

transmission protocol 44

TSTAT macro 33, 44, 48, 52, 54

two-byte code 18

## U

UCS-2 18

Unicode

- converting to 118
- defining sort information 131
- normalization 19

Unicode characters

- defining 130

Unicode encoding

- on BS2000/OSD interfaces 43

Unicode support 18

Unicode variant 18

user default character set 32, 114

- assign 114

- assigning 46

UTF 199

UTF-16 18

UTF-8 18

UTF-8 strings

- outputting length 104

UTFE 18, 199

UTFE strings

- outputting length 104

UTF-EBCDIC 18

## V

V format 101

VTSU 30

VTSU control block 32

VTSU operating parameters 35, 139, 141

VTSU special routines 35, 141

## W

WCP1252

- Windows character set 137

Windows character set

- WCP1252 137

## Y

YINQUIRE macro 33, 44, 50, 53





## Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format *...@ts.fujitsu.com*.

The Internet pages of Fujitsu Technology Solutions are available at

<http://ts.fujitsu.com/...>

and the user documentation at <http://manuals.ts.fujitsu.com>.

Copyright Fujitsu Technology Solutions, 2009

## Hinweise zum vorliegenden Dokument

Zum 1. April 2009 ist Fujitsu Siemens Computers in den alleinigen Besitz von Fujitsu übergegangen. Diese neue Tochtergesellschaft von Fujitsu trägt seitdem den Namen Fujitsu Technology Solutions.

Das vorliegende Dokument aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche Produktversion.

Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.

Kontakt- und Supportadressen werden nun von Fujitsu Technology Solutions angeboten und haben die Form *...@ts.fujitsu.com*.

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter <http://de.ts.fujitsu.com/...>, und unter <http://manuals.ts.fujitsu.com> finden Sie die Benutzerdokumentation.

Copyright Fujitsu Technology Solutions, 2009