



IBM System i™

Session: 25CE

# What's With These ASCII, EBCDIC, Unicode CCSIDs?

Bruce Vining  
Session: 510061

*i want stress-free IT.*  
*i want control.*  
*i want an **i**.*

8 Copyright IBM Corporation, 2007. All Rights Reserved.  
This publication may refer to products that are not currently available in your country. IBM makes no commitment to make available any products referred to herein.

# Abstract

In today's business world there is a growing need to exchange data with other users that might be working in different languages and environments.

This might involve using Unicode to accept and display Russian and Japanese data from a 5250 RPG application, or general data that needs to be received or sent in batch to an AIX application.

This session covers how to use built-in facilities of i5/ OS to work with other systems using encodings such as ASCII, EBCDIC, and Unicode. Samples are provided in RPG, COBOL, C and CL.

By the end of this session, attendees will be able to:

1. Convert data using the iconv API.
2. Support Unicode in a 5250 environment.
3. Support Unicode in a DB2 environment.

# Lets start with some terms

- Character Set – a collection of elements used to represent textual information (e.g. 0-9, a-z, A-Z, .,;,:!/?/\_” @#\$%^&\*()+={}~` ... )
  - A Character Set generally supports more than one language – e.g. Latin-1 Character Set supports all Western European languages
- Code Page – (AKA Code set)
  - where each character in a character set is assigned a numerical representation (often used interchangeably with character set – e.g. charset in HTML)
- CCSID
  - a unique number (0-65535) used by IBM to uniquely identify a Coded Character Set and a Codepage.

# Example of an EBCDIC code page

HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000								¼ NF040000	0 ND100000
-1			/ SP120000		a LA010000	j LJ010000			A LA020000	J LJ020000		1 ND010000
-2					b LB010000	k LK010000	s LS010000		B LB020000	K LK020000	S LS020000	2 ND020000
-3					c LC010000	l LL010000	t LT010000		C LC020000	L LL020000	T LT020000	3 ND030000
-4					d LD010000	m LM010000	u LU010000		D LD020000	M LM020000	U LU020000	4 ND040000
-5					e LE010000	n LN010000	v LV010000		E LE020000	N LN020000	V LV020000	5 ND050000
-6					f LF010000	o LO010000	w LW010000		F LF020000	O LO020000	W LW020000	6 ND060000
-7					g LG010000	p LP010000	x LX010000		G LG020000	P LP020000	X LX020000	7 ND070000
-8					h LH010000	q LQ010000	y LY010000		H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9				± SA020000	i LI010000	r LR010000	z LZ010000		I LI020000	R LR020000	Z LZ020000	9 ND090000
-A			½ NF010000	: SP130000								
-B	· SP110000	¢ SC030000	£ SP100000	# SM010000								
-C		* SM040000	% SM020000	@ SM050000								
-D	( SP060000	) SP070000	_ SP080000	' SP050000								
-E	+ SA010000	; SP140000		- SA040000								
-F		¢ SC040000	? SP150000	" SP040000								Ⓢ

Fixed Code Points

Changeable Code Points

Examples of Characters that do change hex values:

#, \$, @, Å

# Example of a ASCII code page

Fixed Code Points

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

Changeable Code Points

# How come so many different code pages in use?

## The codepage problem exists in both ASCII and EBCDIC

- EBCDIC

- 10 different code pages to support Latin based script (English, French, German etc
  - 37, 297, 500 etc
- 1 to support Greek (plus out of date ones)
- 1 to support Russian (plus out of date ones)
- etc

- ASCII

- 2 code pages to support Latin based scripts
  - 819 for ISO (8859-1) and 1252 for Windows
- 1 to support Greek (plus out of date ones)
- 1 to support Russian (plus out of date ones)
- etc

# CCSID Considerations

- Coded Character Set Identifiers (CCSIDs)
- CCSIDs are used to define a method of assigning and preserving the meaning and rendering of characters through various stages of processing and interchange.
- CCSID support is particularly important when:
  - Converting between encoding schemes (ASCII, EBCDIC, Unicode)
  - Multiple national language versions, keyboards, and display stations are installed on i5/OS.
  - Multiple System i servers are sharing data between systems with different national language versions.
  - The correct keyboard support for a language is not available when you want to encode data in another language.
- i5/OS supports a large set of CCSIDs.
- i5/OS documents which pre-defined CCSID mappings it supports (which CCSIDs a given CCSID can be mapped to)
  - Example: CCSID 00037 can be mapped to about 100 other CCSIDs
  - Some CCSIDs only map to a few other CCSIDs.
- To avoid needing to assign a CCSID to every object, set the CCSID at the system level.

# Common CCSID Values Defined on i5/OS

<b>CCSID</b>	<b>Char Set</b>	<b>Description</b>
00037	697	US, Canada, Netherlands, Portugal, Brazil, New Zealand, Australia, others
00256	697	Netherlands
00273	697	Austria, Germany
00277	697	Denmark, Norway
00278	697	Finland, Sweden
00280	697	Italy
00284	697	Spanish (Latin America)
00285	697	United Kingdom
00290	1172	Japanese
00297	697	France
00937	1175	Chinese Simplified
01025	1150	Russian
.....		

Note that the Western European languages share the same Character Set

# Data Integrity Problems

- Whenever data needs to be converted to a different CCSID and that CCSID has a different character set, the characters in the original CCSID data that do not exist in the destination CCSID will be replaced or substituted
  - Enforced subset match
  - Best fit
  - Round trip
- Conversion is done character by character so not all characters in a field may be changed/lost

## CCSID Example #1: Data integrity is not maintained

- Data integrity may not be maintained using CCSID 65535 across languages. This CCSID is not recommended because it turns off automatic conversion.
- Example showing the purpose of maintaining data integrity.
- An application is being used by different language users. A database file created by a U.S. user contains a dollar sign and is read by a user in the United Kingdom and in Denmark. If the application does not assign CCSID tags that are associated with the data to the file, users see different characters.

Country	Keyboard Type	Code page	CCSID	Code point	Character
U.S.	USB	037	65535	X'5B'	\$
U.K.	UKB	285	65535	X'5B'	£
Denmark	DMB	277	65535	X'5B'	Å

## CCSID Example #2: Data integrity is maintained

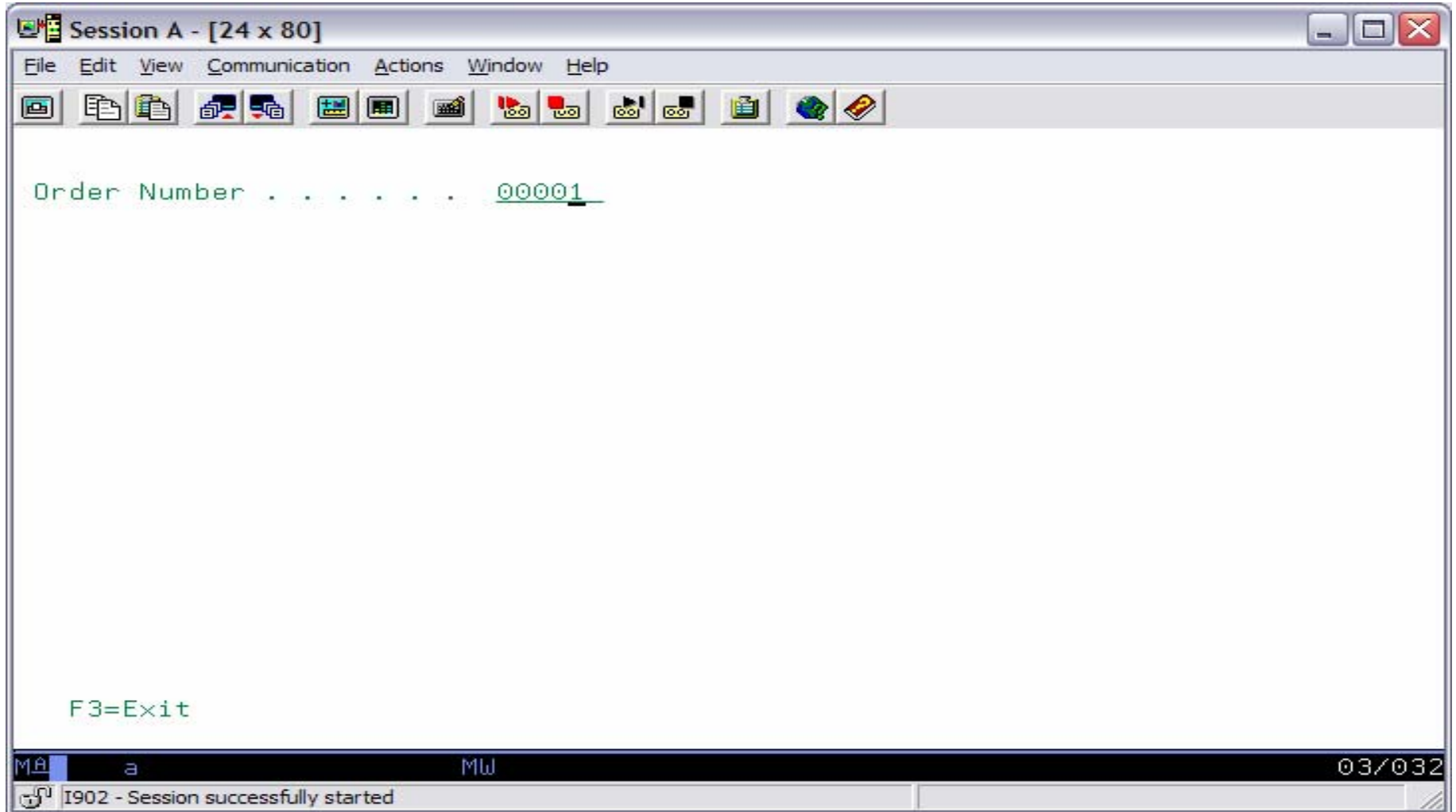
- Data integrity is maintained by using CCSID tags.
- If the application assigns a CCSID associated with the data to a file, the application can use i5/OS CCSID support to maintain the integrity of the data. When the file is created with CCSID 037, the user in the United Kingdom (job CCSID 285) and the user in Denmark (job CCSID 277) see the same character. Database management takes care of the mapping.

Country	Keyboard Type	Code page	CCSID	Code point	Character
U.S.	USB	037	00037	X'5B'	\$
U.K.	UKB	285	00285	X'4A'	\$
Denmark	DMB	277	00277	X'67'	\$

# So what is Unicode?

- Unicode is the **universal character encoding standard** used for representation of text for computer processing.
- Can be used to store & process all significant current & past languages
- Unicode provides a unique hex encoded number for every character,
  - no matter what the platform, program or language
- The Unicode Standard has been adopted by industry leaders
  - Apple, HP, IBM, Microsoft, Oracle, SAP, Sun, Sybase, Unisys
  - many others.
- Unicode is required by web users and modern standards
  - XML, Java, ECMAScript (JavaScript), LDAP, CORBA 3.0, WML

# Sample Interactive Ship To Display



# Sample Interactive Ship To Display Using English and CCSID 37

```
Session B - [24 x 80]
File Edit View Communication Actions Window Help
Ship To Information
Company . . . . . ABC Company
Contact . . . . . Bruce Vining
Status . . . . . 0
Ship to address . . . . 3605 Highway 52 North
Rochester, MN 55901
Part No Part Description Quantity
1 Hammer 13
2 Nail 4
3 Wrench 7
F3=Exit
MA b
I902 - Session successfully started
```

# Sample Interactive Ship To Physical File DDS

ORDER (Order Summary):

				UNIQUE
R	ORDREC			
	ORDNO	5	0	
	ORDSTS	1		
	COMPANY	40		
	CONTACT	40		
	ADDR1	40		
	ADDR2	40		
K	ORDNO			

INVEN (Inventory Descriptions):

				UNIQUE
R	INVREC			
	PARTNO	5	0	
	PARTDESC	40		
K	PARTNO			

ORDDDET (Order Detail):

				UNIQUE
R	ORDDEC			
	ORDNO	R		REFFLD(ORDREC/ORDNO ORDER)
	PARTNO	R		REFFLD(INVREC/PARTNO INVEN)
	ORDERQTY	6	0	
K	ORDNO			
K	PARTNO			

# Sample Interactive Ship To Display File DDS

CF03(03)

\* Command key prompts

R FMT1

23 4'F3=Exit'

\* Prompt for Order Number

R PROMPT

50

ORDNO R

3 2'Order Number . . . . .'  
I 3 28REFFLD(ORDREC/ORDNO ORDER)

22 2'Incorrect Order Number'

23 4'F3=Exit'

\* Subfile for parts ordered

R SFLRCD

SFL

PARTNO R

O 12 4REFFLD(ORDDEC/PARTNO ORDDET)  
EDTWRD(' , ')

PARTDESC R

O 12 12REFFLD(INVREC/PARTDESC INVEN)

ORDERQTY R

O 12 65EDTWRD(' , ')  
REFFLD(ORDDEC/ORDERQTY ORDDET)

# Sample Interactive Ship To Display File DDS

\* Subfile control and main display

```

      R SFLCTL                                SFLCTL(SFLRCD)
                                           SFLSIZ(100)
                                           SFLPAG(9)
N25   SFLDSPCTL
                                           OVERLAY
      21   SFLDSP
      25   SFLCLR
                                           1 28'Ship To Information'
                                           3  2'Company . . . . . '
      COMPANY      R      O  3 28REFFLD(ORDREC/COMPANY ORDER)
                                           4  2'Contact . . . . . '
      CONTACT      R      O  4 28REFFLD(ORDREC/CONTACT ORDER)
                                           6  2'Status . . . . . '
      ORDSTS       R      O  6 28REFFLD(ORDREC/ORDSTS ORDER)
                                           8  2'Ship to address . . . . . '
      ADDR1        R      O  8 28REFFLD(ORDREC/ADDR1 ORDER)
      ADDR2        R      O  9 28REFFLD(ORDREC/ADDR2 ORDER)
                                           11 4'Part No'
                                           11 12'Part Description'
                                           11 65'Quantity'

```

# Sample ILE RPG Interactive Program Files and Working Fields

```
fshiptodspfcf    e                workstn sfile(sflrcd:RelRecNbr)
forder          if    e                k disk
forddet         if    e                k disk
finven          if    e                k disk

dRelRecNbr      s                4  0
```

# Sample ILE RPG Interactive Program

```
* Prompt for order number until Command Key 3
c           dow           *in03 <> '1'
c           exfmt         prompt

* Get summary order information if it exists
c   ordno      chain      ordrec              50
c           if           *in50 = *on
c           iter
c           endif

* Get detail order information
c   ordno      setll      orddec
c   ordno      reade      orddec              51
c           dow           *in51 = *off

* Get translated part descriptions
c   partno     chain      invrec
c           eval      RelRecNbr += 1
c           write     SflRcd
c   ordno      reade      orddec              51
c           enddo
```

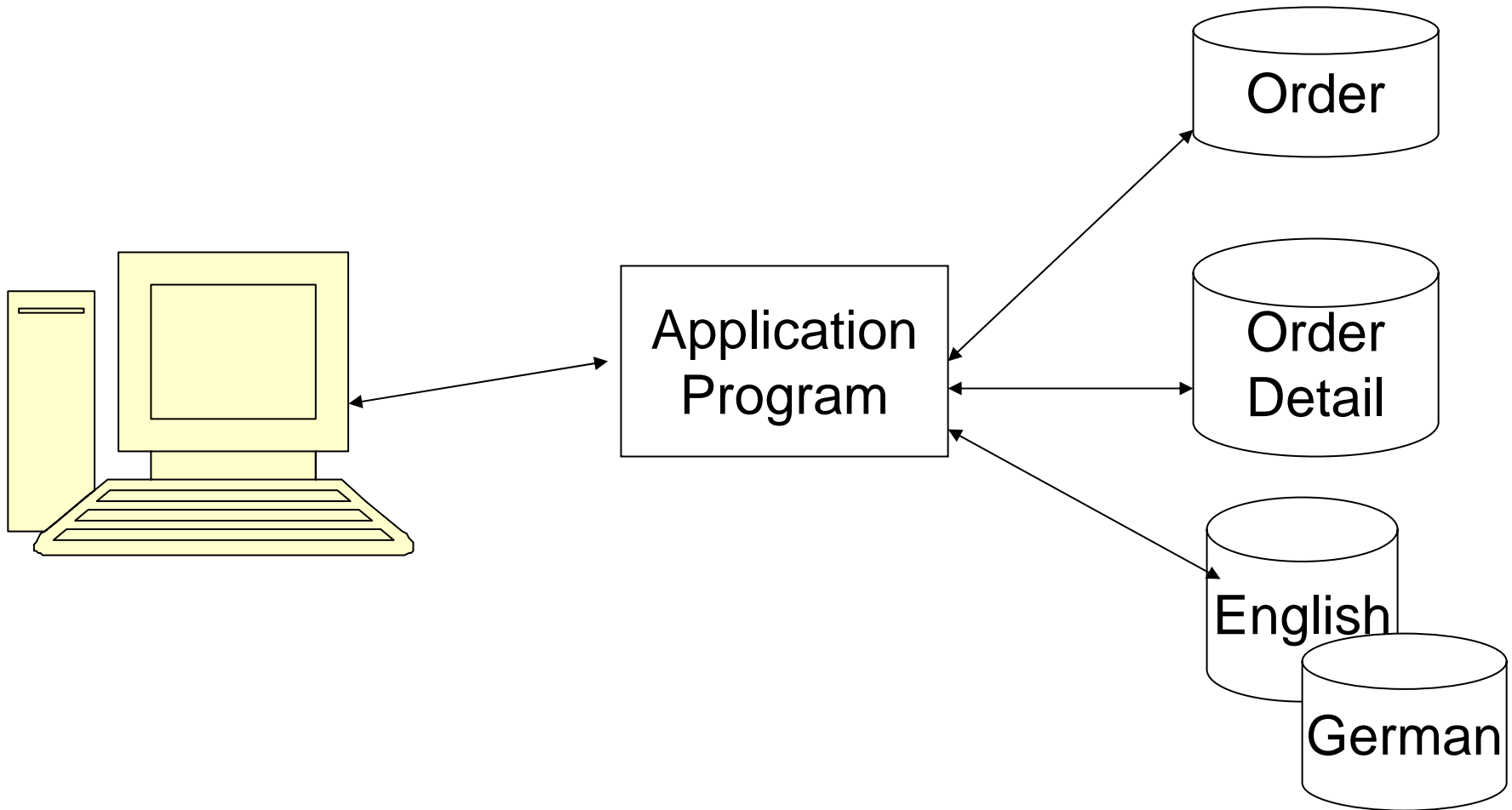
# Sample ILE RPG Interactive Program

```
* Write the display
c           write      Fmt1
c           if         RelRecNbr > 0
c           eval      *in21 = *on
c           endif
c           exfmt      SflCtl
c           eval      *in21 = *off

* Clear the subfile and return to prompt for order number
c           eval      *in25 = *on
c           write      SflCtl
c           eval      *in25 = *off
c           eval      RelRecNbr = 0
c           enddo

c           eval      *inlr = *on
c           return
```

# Approach to Inventory Parts Descriptions



# Sample Interactive Ship To Display Using German Part Descriptions and CCSID 37

```
Session A - [24 x 80]
File Edit View Communication Actions Window Help
Ship To Information
Company . . . . . ABC Company
Contact . . . . . Bruce Vining
Status . . . . . 0
Ship to address . . . . 3605 Highway 52 North
Rochester, MN 55901
Part No Part Description Quantity
1 Hammer 13
2 Nagel 4
3 Schraubenschlüssel 7
F3=Exit
MA a MW 01/001
I902 - Session successfully started
```

# Sample Interactive Ship To Display Using German Part Descriptions and Cyrillic Company Information – Display configured as Cyrillic (CCSID 1025)

```

Session B - [24 x 80]
File Edit View Communication Actions Window Help

Ship To Information

Company . . . . . Россия Shipping
Contact . . . . . Павлу Ноздрину

Status . . . . . 0

Ship to address . . . . Краснопресненская наб., д. 18
                        123317 г. Москва, Россия,

Part No Part Description                               Quantity
  1 Hammer                                           13
  2 Nagel                                             4
  3 Schraubenschl [REDACTED]                          7

Not all of part number 3's description
displays as the character does not
exist in CCSID 1025

Same effect if user needs to see both
German and Cyrillic orders in same
session.

F3=Exit

M&A b
I902 - Session successfully started
  
```

# The Answer is Unicode and an Emulator such as System i Access for Web

Session A - X1519P4.RCHLAND.IBM.COM [X1519P4.RCHLAND.IBM.COM] - Microsoft Internet Explorer

User: vining System: X1519P4.RCHLAND.IBM.COM

## iSeries Access for Web

Ship To Information

Company ..... Россия Shipping  
 Contact ..... Павлу Ноздрину

Status ..... 0

Ship to address .... Краснопресненская наб., д. 18  
 123317 г. Москва, Россия,

Part No	Part Description	Quantity
1	Hammer	13
2	Nagel	4
3	Schraubenschlüssel	7

F3=Exit

Attention	Refresh Screen	Field Exit	Page Up	Enter
System Request	Stop Session	Reset	Page Down	

IBM | iSeries | Service 5.4.0.05-191.S124678

Done Internet

# How about Russian, Chinese, and German?

## On the same panel, or different orders on same device at different times

Session A - X1519P4.RCHLAND.IBM.COM [X1519P4.RCHLAND.IBM.COM] - Microsoft Internet Explorer

User: vining System: X1519P4.RCHLAND.IBM.COM

### Ship To Information

Company ..... Russian & China world coverage  
 Contact ..... 成儒先生

Status ..... H

Ship to address .... Краснопесненская наб., д. 18  
 123317 г. Москва, Россия,

Part No	Part Description	Quantity
1	Hammer	13
2	Nagel	4
3	Schraubenschlüssel	7

F3=Exit

Attention	Refresh Screen	Field Exit	Page Up	Enter
System Request	Stop Session	Reset	Page Down	

IBM | iSeries | Service 5.4.0.05-191.S124678

# Only Database Definition Changes to Support Unicode for This Example

- ORDER file:

		UNIQUE
R	ORDREC	
	ORDNO	5 0
	ORDSTS	1
	COMPANY	40G CCSID (13488 20)
	CONTACT	40G CCSID (13488 20)
	ADDR1	40G CCSID (13488 20)
	ADDR2	40G CCSID (13488 20)
K	ORDNO	

No need to change ORDSTS as Status code does not need to be internationalized

Other character based fields are changed to Graphic with CCSID 13488 and a display length of 40 bytes (20 x 2)

- INVEN file:

		UNIQUE
R	INVREC	
	PARTNO	5 0
	PARTDESC	40G CCSID (13488 20)
K	PARTNO	

- ORDER DETAIL file:

		UNIQUE
R	ORDDEC	
	ORDNO	R REFFLD (ORDREC/ORDNO ORDER)
	PARTNO	R REFFLD (INVREC/PARTNO INVEN)
	ORDERQTY	6 0
K	ORDNO	
K	PARTNO	

Do need to recompile \*DSPF and RPG application to pick up new definitions

# More Complex Programs Most Likely Need Changes

- Working variable definitions
- ILE RPG PTFs to help unlike data type operations:
  - Eval
  - If, When, DOW, DOU
  - Inz
  
  - V5R3: SI24532
  - V5R4: SI26312
  - V5R4: SI25232 if compiling to V5R3 release
- but some areas to watch out for:
  - Concatenation
  - %scan
  - Same named fields on I specs
  - Parameter passing

# Need more control?

- There are many ways within i5/OS to convert data from one CCSID to another CCSID:
  - Copy To/From Import File
  - Logical Files
  - Copy File
  - etc
- But what if you want to directly control the conversion within your application program?
  - Direct communications with another system
  - Utilities don't meet exact requirements
  - etc
  
  - Use iconv – a system API for data conversion
  - iconv is what's effectively used by the system under the covers...

# iconv

## Prototypes for common routines

```
h dftactgrp(*no)
```

```

dSetConvert      pr          10i 0
d InputCCSID     10i 0 value
d OutputCCSID    10i 0 value

dConvert         pr          10i 0
d Input          *    value
d Len_Input      10i 0 value

dEndConvert      pr          10i 0 extproc('iconv_close')
d ConvDesc              value like(cd)

```

Some common functions to help you on your way.

SetConvert: what CCSID do you want to convert from and to

Convert: the name says it all and can be called as many times as you want

EndConvert: for when you're done using Convert

# iconv

## Working variables

dcd	ds		
d cdBins		10i 0	dim(13)
dInput_Variable	s	50	inz('Some variable data')
dInput_Number	s	10i 0	inz(101355)
dOutput_Value	s	4096	
dLen_Output	s	10i 0	
dRtnCde	s	10i 0	

# iconv

## Specify what CCSID to convert from and to

- \* Set our working CCSID to 37 for this example and ask for
- \* conversion to UTF 16

```
c          eval          RtnCde = SetConvert(37 :1200)
c          if            RtnCde = 0
```

# iconv

## Convert a character variable

- \* Convert an EBCDIC field (note: don't trim input Unicode fields when
- \* using a character based definition (as in this example) as a
- \* leading/trailing x'40' can easily be real data in Unicode - trim
- \* would be OK if the field is defined as UCS-2 (datatype C))

```
c          eval          RtnCde = Convert(%addr(Input_Variable)
c                                     :%len(%trimr(Input_Variable)))
c
c          if            RtnCde = -1
c      'Text Error' dsply
c          else
```

- \* Output\_Value now contains the converted field with a length of
- \* Len\_Output bytes

```
c          endif
```

# iconv

## Convert a numeric value

\* Convert a numeric variable (101355)

```
c          eval          Input_Variable = %char(Input_Number)
c          eval          RtnCde = Convert(%addr(Input_Variable)
c                                :%len(%char(Input_Number)))
c          if            RtnCde = -1
c      'Number Error'dsply
c          else
c
c      * Output_Value now contains the converted field with a length of
c      * Len_Output bytes
c
c          endif
```

# iconv

## When you are done

\* Close the cd after all conversions are done

```
c          eval          RtnCde = EndConvert(cd)
c          endif
c          eval          *inlr = '1'
c          return
```

# iconv

## SetConvert common routine

```

pSetConvert      b
dSetConvert      pi          10i 0
d InputCCSID     10i 0 value
d OutputCCSID    10i 0 value

dConvertOpen     pr          52a  extproc('QtqIconvOpen')
d ToCode         *          value
d FromCode       *          value

dToCode          ds
d ToCCSID        10i 0
d ToConvAlt      10i 0 inz(0)
d ToSubAlt       10i 0 inz(0)
d ToStateAlt     10i 0 inz(0)
d ToLenOpt       10i 0 inz(0)
d ToErrOpt       10i 0 inz(0)
d TReserved      8        inz(*allx'00')

dFromCode        ds
d FromCCSID      10i 0
d FromConvAlt    10i 0 inz(0)
d FromSubAlt     10i 0 inz(0)
d FromStateAlt   10i 0 inz(0)
d FromLenOpt     10i 0 inz(0)
d FromErrOpt     10i 0 inz(0)
d FReserved      8        inz(*allx'00')

```

# iconv

## SetConvert common routine

```
c          eval          FromCCSID = InputCCSID
c          eval          ToCCSID = OutputCCSID
c          eval          cd = ConvertOpen( %addr(ToCode)
c                                     :%addr(FromCode))
c          if            cdBins(1) = -1
c          'Open Error'  dsply
c          return        -1
c          else
c          return        0
c          endif
pSetConvert          e
```

# iconv

## Convert common routine

```

pConvert          b
dConvert          pi          10i 0
d Input_Pointer   *          value
d Input_Length    10i 0      value

diconv           pr          10i 0 extproc('iconv')
d ConvDesc        *          value like(cd)
d InputData       *          value
d InputDataLeft   10i 0
d OutputData      *          value
d OutputDataLeft  10i 0

dOutBufPtr        s          *
dInBytesLeft      s          10i 0
dOutBytesLeft     s          10i 0

```

# iconv

## Convert common routine

- \* reset InBytesLeft, OutBytesLeft, and OutBufPtr each time as iconv
- \* API updates these values

```
c          eval          InBytesLeft = Input_Length
c          eval          OutBytesLeft = %len(Output_Value)
c          eval          OutBufPtr = %addr(Output_Value)
c          eval          RtnCde = iconv( cd
c                               :%addr(Input_Pointer)
c                               :InBytesLeft
c                               :%addr(OutBufPtr)
c                               :OutBytesLeft)
c          if            RtnCde = -1
c      'Conv Error' dsply
c          return        -1
c          else
c          eval          Len_Output = %len(Output_Value)
c                               - OutBytesLeft
c          return        0
c          endif
pConvert      e
```

# References

- System i globalization home page
  - <http://www-03.ibm.com/servers/eserver/series/software/globalization/>
- List & view of IBM codepages
  - <http://www-03.ibm.com/servers/eserver/series/software/globalization/codepages.html>
- G11N Api's
  - <http://publib.boulder.ibm.com/infocenter/series/v5r4/index.jsp?topic=/apis/nls1.htm>
- Unicode site
  - <http://www.unicode.org>

# Examples in Other Languages

# Ship to Application - COBOL

PROCESS CVTPICNGRAPHIC.

IDENTIFICATION DIVISION.  
PROGRAM-ID. SHIPTOCBL.

ENVIRONMENT DIVISION.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.

```
SELECT Ship-To-DSPF ASSIGN TO WORKSTATION-SHIPTODSPF
      ORGANIZATION IS TRANSACTION
      ACCESS MODE IS DYNAMIC
      RELATIVE KEY IS RelRecNbr.
SELECT Order-File ASSIGN TO DATABASE-ORDER
      ORGANIZATION IS INDEXED
      RECORD KEY IS OrdNo OF OrdRec
      ACCESS MODE IS DYNAMIC.
SELECT Order-Detail ASSIGN TO DATABASE-ORDDDET
      ORGANIZATION IS INDEXED
      RECORD KEY IS EXTERNALLY-DESCRIBED-KEY
      ACCESS MODE IS DYNAMIC.
SELECT Inven-File ASSIGN TO DATABASE-INVEN
      ORGANIZATION IS INDEXED
      RECORD KEY IS PartNo OF InvRec
      ACCESS MODE IS DYNAMIC.
```

# Ship to Application - COBOL

```
DATA DIVISION.
FILE SECTION.
FD  Ship-To-DSPF.
01  Ship-To-DSPF-Records.
    COPY DDS-ALL-FORMATS OF SHIPTODSPF.
FD  Order-File.
01  Order-File-Records.
    COPY DDS-ALL-FORMATS OF ORDER.
FD  Order-Detail.
01  Order-Detail-Records.
    COPY DDS-ALL-FORMATS OF ORDDDET.
FD  Inven-File.
01  Inven-File-Records.
    COPY DDS-ALL-FORMATS OF INVEN.

WORKING-STORAGE SECTION.
01  Prompt-I-DS.
    COPY DDS-PROMPT-I OF SHIPTODSPF.
01  SFLRCD-O-DS.
    COPY DDS-SFLRCD-O OF SHIPTODSPF.
01  SFLCTL-I-DS.
    COPY DDS-SFLCTL-I OF SHIPTODSPF.
01  SFLCTL-O-DS.
    COPY DDS-SFLCTL-O OF SHIPTODSPF.

01  RelRecNbr                PIC  9(4) VALUE 0.
```

# Ship to Application - COBOL

PROCEDURE DIVISION.

MAIN-LINE.

OPEN I-O Ship-To-DSPF.

OPEN INPUT Order-File, Order-Detail, Inven-File.

MOVE ZEROS TO PartNo of OrdDec

PERFORM UNTIL IN03 OF SFLCTL-I-DS EQUAL B"1"

WRITE Ship-To-DSPF-Records FORMAT "PROMPT"

READ Ship-To-DSPF INTO Prompt-I-DS

IF IN03 OF Prompt-I-DS EQUAL B"1"

GO TO Done

END-IF

MOVE OrdNo OF Prompt-I-DS TO OrdNo of OrdRec,  
OrdNo of OrdDec

READ Order-File INVALID KEY MOVE B"1" TO IN50

END-READ

# Ship to Application - COBOL

```
IF IN50 NOT EQUAL B"1"  
  MOVE CORR OrdRec TO SflCtl-O OF SflCtl-O-DS  
  MOVE 0 TO RelRecNbr  
  MOVE B"1" TO IN25 OF SflCtl-O-DS  
  WRITE Ship-To-DSPF-Records FROM  
    SflCtl-O OF SflCtl-O-DS FORMAT IS "SFLCTL"  
  MOVE B"0" TO IN25 OF SflCtl-O-DS  
  MOVE ZEROS TO PartNo OF OrdDec  
  START Order-Detail KEY NOT LESS THAN  
    EXTERNALLY-DESCRIBED-KEY  
  READ Order-Detail NEXT  
  PERFORM WITH TEST BEFORE UNTIL  
    OrdNo OF OrdDec NOT EQUAL OrdNo OF Prompt-I-DS  
  MOVE PartNo OF OrdDec TO PartNo of InvRec  
  READ Inven-File  
    KEY IS PartNo OF InvRec  
  ADD 1 TO RelRecNbr  
  MOVE CORR OrdDec TO SflRcd-O OF SflRcd-O-DS  
  MOVE CORR InvRec TO SflRcd-O OF SflRcd-O-DS  
  WRITE SUBFILE Ship-To-DSPF-Records FROM SflRcd-O-DS  
    FORMAT IS "SFLRCD"  
  READ Order-Detail NEXT  
    AT END MOVE ZEROS TO PartNo OF OrdDec  
  END-READ  
END-PERFORM
```

# Ship to Application - COBOL

```
IF RelRecNbr > 0
    MOVE B"1" TO IN21 OF SflCtl-O-DS
ELSE
    MOVE B"0" TO IN21 OF SflCtl-O-DS
END-IF
WRITE Ship-To-DSPF-Records FROM SflCtl-O-DS
    FORMAT IS "SFLCTL"
READ Ship-To-DSPF INTO SflCtl-I OF SflCtl-I-DS
END-IF
END-PERFORM.
```

Done.

```
CLOSE Ship-To-DSPF, Order-File, Order-Detail, Inven-File.
STOP RUN.
```

# iconv - COBOL

```
PROCESS NOMONOPRC.
```

```
IDENTIFICATION DIVISION.
```

```
PROGRAM-ID. CVTCBL.
```

```
DATA DIVISION.
```

```
WORKING-STORAGE SECTION.
```

```
01  Conv-Desc                                GLOBAL.  
    05  cdBins                               PIC S9(9) BINARY OCCURS 13.  
01  Input-Variable                           PIC X(50)  
                                           VALUE "Some variable data".  
01  Input-Number                             PIC S9(9) BINARY  
                                           VALUE 101355.  
01  Length-Input                             PIC S9(9) BINARY.  
01  Output-Value                             PIC X(4096)          GLOBAL.  
01  Length-Output                           PIC S9(9) BINARY GLOBAL.  
01  Rtn-Cde                                 PIC S9(9) BINARY.
```

# iconv - COBOL

```
PROCEDURE DIVISION.
```

```
MAIN-LINE.
```

- \* Set our working CCSID to 37 for this example and ask for
- \* conversion to UTF 16

```
    CALL "SetConvert" USING BY VALUE 37,  
                           BY VALUE 1200,  
                           RETURNING Rtn-Cde.
```

```
    IF Rtn-Cde = 0
```

- \* Convert an EBCDIC field (note: don't trim input Unicode fields
- \* when using a character based definition (as in this example)
- \* as a leading/trailing x'40' can easily be real data in Unicode
- \* leading/trailing x'40' can easily be real data in Unicode -
- \* trim would be OK if the field is defined as UCS-2 (National))

```
    COMPUTE Length-Input =  
        FUNCTION LENGTH( FUNCTION TRIMR( Input-Variable))  
    CALL "Convert" USING BY VALUE  
                        ADDRESS OF Input-Variable,  
                        BY VALUE      Length-Input,  
                        RETURNING     Rtn-Cde
```

```
    IF Rtn-Cde = -1  
        DISPLAY "Text Error"  
    END-IF
```

- \* Output-Value now contains the converted field with a length of
- \* Length-Output bytes

# iconv - COBOL

\* Convert a numeric variable

```
MOVE Input-Number TO Input-Variable
MOVE FUNCTION TRIML( Input-Variable, "0")
  TO Input-Variable
COMPUTE Length-Input =
  FUNCTION LENGTH( FUNCTION TRIMR( Input-Variable))
CALL "Convert" USING BY VALUE
                        ADDRESS OF Input-Variable,
                        BY VALUE      Length-Input,
                        RETURNING     Rtn-Cde

IF Rtn-Cde = -1
  DISPLAY "Number Error"
END-IF
```

\* Output-Value now contains the converted field with a length of

\* Length-Output bytes

# iconv - COBOL

```
ELSE
```

```
    DISPLAY "SetConvert error"
```

```
END-IF
```

```
* Close the cd after all conversions are done
```

```
CALL LINKAGE PRC "iconv_close" USING  
    BY REFERENCE Conv-Desc,  
    RETURNING      Rtn-Cde.
```

```
STOP RUN.
```

# iconv - COBOL

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. "SetConvert".
```

```
DATA DIVISION.
```

```
WORKING-STORAGE SECTION.
```

```
COPY QTQICONV OF QSYSINC-QCBLLESRC REPLACING  
  ==01 QTQCODE== BY ==01 QTQCODE IS TYPEDEF==.
```

```
01 Rtn-Cde                PIC S9(9) BINARY.  
01 From-Code.  
   05 From-Environment    TYPE QTQCODE.  
01 To-Code.  
   05 To-Environment      TYPE QTQCODE.
```

```
LINKAGE SECTION.
```

```
01 Input-CCSID            PIC S9(9) BINARY.  
01 Output-CCSID           PIC S9(9) BINARY.
```

# iconv - COBOL

```
PROCEDURE DIVISION USING BY VALUE Input-CCSID,  
                        BY VALUE Output-CCSID,  
                        RETURNING Rtn-Cde.
```

```
MAIN-LINE.
```

```
    MOVE LOW-VALUES TO To-Code.  
    MOVE LOW-VALUES TO From-Code.  
    MOVE Input-CCSID TO CCSID OF From-Code.  
    MOVE Output-CCSID TO CCSID OF To-Code.  
    CALL LINKAGE PRC "QtqIconvOpen" USING  
                BY REFERENCE To-Code,  
                BY REFERENCE From-Code,  
                RETURNING Conv-Desc.
```

```
    IF cdBins(1) = -1  
        DISPLAY "Open error"  
        MOVE -1 TO Rtn-Cde
```

```
    ELSE
```

```
        MOVE 0 TO Rtn-Cde
```

```
    END-IF
```

```
    GOBACK.
```

```
END PROGRAM "SetConvert".
```

# iconv - COBOL

```
IDENTIFICATION DIVISION.  
PROGRAM-ID. "Convert".
```

```
DATA DIVISION.
```

```
WORKING-STORAGE SECTION.
```

```
01  Rtn-Cde                PIC S9(9) BINARY.  
01  Output-Buffer-Pointer  POINTER.  
01  Input-Bytes-Left       PIC S9(9) BINARY.  
01  Output-Bytes-Left     PIC S9(9) BINARY.
```

```
LINKAGE SECTION.
```

```
01  Input-Pointer          POINTER.  
01  Input-Length          PIC S9(9) BINARY.
```

# iconv - COBOL

```
PROCEDURE DIVISION USING BY VALUE Input-Pointer,  
                        BY VALUE Input-Length,  
                        RETURNING Rtn-Cde.
```

```
MAIN-LINE.
```

- \* Reset Input-Bytes-Left, Output-Bytes-Left, and
- \* Output-Buffer-Pointer each time as iconv updates these values

```
MOVE Input-Length TO Input-Bytes-Left.  
MOVE LENGTH OF Output-Value TO Output-Bytes-Left.  
SET Output-Buffer-Pointer TO ADDRESS OF Output-Value.  
CALL LINKAGE PRC "iconv" USING  
    BY VALUE      Conv-Desc,  
    BY VALUE      ADDRESS OF Input-Pointer,  
    BY REFERENCE Input-Bytes-Left,  
    BY VALUE      ADDRESS OF  
                  Output-Buffer-Pointer,  
    BY REFERENCE Output-Bytes-Left,  
    RETURNING     Rtn-Cde.
```

```
IF Rtn-Cde = -1  
    DISPLAY "Conv Error"  
ELSE  
    COMPUTE Length-Output = LENGTH OF Output-Value -  
                          Output-Bytes-Left  
  
    MOVE 0 TO Rtn-Cde  
END-IF  
GOBACK.
```

```
END PROGRAM "Convert".
```

```
END PROGRAM CVTCBL.
```

# iconv - CL

```
PGM
DCL      VAR(&FROMCCSID) TYPE(*INT) VALUE(37)
DCL      VAR(&TOCCSID) TYPE(*INT) VALUE(1200)
DCL      VAR(&RC) TYPE(*INT)
DCL      VAR(&MOVESIZE) TYPE(*INT) VALUE(32)
DCL      VAR(&INBYTELEFT) TYPE(*INT)
DCL      VAR(&OUTBYTLEFT) TYPE(*INT)
DCL      VAR(&INPUTVAR) TYPE(*CHAR) VALUE('Some +
      variable data')
DCL      VAR(&INPUTNUM) TYPE(*INT) VALUE(101355)
DCL      VAR(&INPUTCHR) TYPE(*CHAR) LEN(6)
DCL      VAR(&INPUTPTR) TYPE(*PTR)
DCL      VAR(&OUTPUTPTR) TYPE(*PTR)
DCL      VAR(&OUTPUTVAR) TYPE(*CHAR) LEN(4096)

DCL      VAR(&CD) TYPE(*CHAR) LEN(52)
DCL      VAR(&CDRC) TYPE(*INT) STG(*DEFINED) DEFVAR(&CD)

DCL      VAR(&FROMCODE) TYPE(*CHAR) LEN(32)
DCL      VAR(&FCCSID) TYPE(*INT) STG(*DEFINED) +
      DEFVAR(&FROMCODE)

DCL      VAR(&TOCODE) TYPE(*CHAR) LEN(32)
DCL      VAR(&TCCSID) TYPE(*INT) STG(*DEFINED) +
      DEFVAR(&TOCODE)
```

# iconv - CL

```
/* Initialize &TOCODE and &FROMCODE to all x'00'1      */
/* and set the appropriate CCSID values                  */
CALLPRC      PRC('_PROPB') PARM((&TOCODE) (X'00' *BYVAL) +
      (&MOVESIZE *BYVAL))

CHGVAR      VAR(&TCCSID) VALUE(&TOCCSID)
CALLPRC      PRC('_PROPB') PARM((&FROMCODE) (X'00' +
      *BYVAL) (&MOVESIZE *BYVAL))

CHGVAR      VAR(&FCCSID) VALUE(&FROMCCSID)

CALLPRC      PRC('QtqIconvOpen') PARM((&TOCODE) +
      (&FROMCODE)) RTNVAL(&CD)

IF          COND(&CDRC = -1) THEN(DO)
SNDPGMMSG   MSG('Open error') TOPGMQ(*EXT)
RETURN
ENDDO
```

# iconv - CL

```
CHGVAR      VAR (&INPUTPTR)  VALUE (%ADDRESS (&INPUTVAR) )
CHGVAR      VAR (&OUTPUTPTR) VALUE (%ADDRESS (&OUTPUTVAR) )
CHGVAR      VAR (&INBYTELEFT) VALUE (18)
CHGVAR      VAR (&OUTBYTLEFT) VALUE (4096)
CALLPRC     PRC ('iconv') PARM ((&CD *BYVAL) (&INPUTPTR) +
                                (&INBYTELEFT) (&OUTPUTPTR) (&OUTBYTLEFT) ) +
                                RTNVAL (&RC)
IF          COND (&RC = -1) THEN (DO)
            SNDPGMMSG  MSG ('Conv error') TOPGMQ (*EXT)
            RETURN
            ENDDO
```

# iconv - CL

```
CHGVAR      VAR (&INPUTCHR)  VALUE (&INPUTNUM)
CHGVAR      VAR (&INPUTPTR)  VALUE (%ADDRESS (&INPUTCHR) )
CHGVAR      VAR (&OUTPUTPTR)  VALUE (%ADDRESS (&OUTPUTVAR) )
CHGVAR      VAR (&INBYTELEFT) VALUE (6)
CHGVAR      VAR (&OUTBYTLEFT) VALUE (4096)
CALLPRC     PRC ('iconv') PARM ((&CD *BYVAL) (&INPUTPTR) +
                                (&INBYTELEFT) (&OUTPUTPTR) (&OUTBYTLEFT)) +
                                RTNVAL (&RC)
IF          COND (&RC = -1) THEN (DO)
            SNDPGMMSG  MSG ('Number error') TOPGMQ (*EXT)
            RETURN
            ENDDO

CALLPRC     PRC ('iconv_close') PARM ((&CD))
ENDPGM
```

# Backup material

# Code Page 37: US, Canada others

HEX DIGITS	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
1ST →												
2ND ↓												
-0	SP10000 ø	SM030000 &	SP100000 -	LO610000 ø	LO620000 Ø	SM190000 °	SM170000 μ	SD150000 ^	SM110000 {	SM140000 }	SM070000 \	ND100000 0
-1	SP300000 ø	LE110000 é	SP120000 /	LE120000 É	LA010000 a	LJ010000 j	SD190000 ~	SC020000 £	LA020000 A	LJ020000 J	SA050000 +	ND010000 1
-2	LA150000 â	LE150000 ê	LA160000 Â	LE160000 Ê	LB010000 b	LK010000 k	LB010000 s	SC050000 ¥	LB020000 B	LK020000 K	LS020000 S	ND020000 2
-3	LA170000 ã	LE170000 ë	LA180000 Ã	LE180000 Ë	LD010000 c	LL010000 l	LT010000 t	SD630000 ·	LC020000 C	LL020000 L	LT020000 T	ND030000 3
-4	LA130000 à	LE130000 è	LA140000 À	LE140000 È	LD010000 d	LM010000 m	LU010000 u	SM520000 ©	LD020000 D	LM020000 M	LU020000 U	ND040000 4
-5	LA110000 á	LI110000 í	LA120000 Á	LI120000 Í	LE010000 e	LN010000 n	LV010000 v	SM240000 §	LE020000 E	LN020000 N	LV020000 V	ND050000 5
-6	LA190000 ä	LI150000 î	LA200000 Ä	LI160000 Ï	LF010000 f	LO010000 o	LW010000 w	SM250000 ¶	LF020000 F	LO020000 O	LW020000 W	ND060000 6
-7	LA270000 å	LI170000 ÿ	LA280000 Å	LI180000 Ï	LG010000 g	LP010000 p	LX010000 x	NF040000 ¼	LG020000 G	LP020000 P	LX020000 X	ND070000 7
-8	LC410000 ç	LI130000 ì	LC420000 Ç	LI140000 Ì	LH010000 h	LQ010000 q	LY010000 y	NF010000 ½	LH020000 H	LQ020000 Q	LY020000 Y	ND080000 8
-9	LI190000 ñ	LB010000 ß	LN200000 Ñ	SD130000 ,	LI010000 i	LR010000 r	LZ010000 z	NF050000 ¾	LI020000 I	LR020000 R	LZ020000 Z	ND090000 9
-A	SC040000 ¢	SP020000 !	SM650000 	SP130000 :	SP170000 «	SM210000 ª	SP030000 ¡	SM060000 [	SP320000 [ (ñv)	ND011000 1	ND021000 2	ND031000 3
-B	SP110000 .	SC030000 \$	SP080000 ,	SM010000 #	SP160000 »	SM200000 º	SP160000 ¸	SM080000 ]	LO150000 ð	LU150000 ú	LO160000 ô	LU160000 û
-C	SA030000 <	SM040000 *	SM020000 %	SM050000 @	LD630000 ð	LA510000 æ	LD620000 Ð	SM150000 -	LO170000 ö	LU170000 ü	LO180000 ò	LU180000 ù
-D	SP060000 (	SP070000 )	SP050000 _	SP050000 '	LY110000 ý	SD410000 .	LY120000 Ý	SD170000 "	LO130000 ò	LU130000 ù	LO140000 ò	LU140000 ù
-E	SA010000 +	SP140000 ;	SA050000 >	SA040000 =	LT630000 þ	LA520000 Æ	LT640000 Ð	SD110000 '	LO110000 ó	LU110000 ú	LO120000 ó	LU120000 ú
-F	SM130000 	SM660000 ¬	SP150000 ?	SP040000 "	SA020000 ±	SC010000 ¤	SM530000 ®	SA070000 x	LO190000 õ	LY170000 ÿ	LO200000 ö	LO000000 @

Code Page 00037

# Code Page 273: German

HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	SP010000	SM000000	SP100000	LOE10000	LOK20000	SM100000	SM170000	SC040000	LA170000	LU170000	LO100000	ND100000
-1	SP300000	LE110000	SP120000	LE120000	LA010000	LJ010000	L8010000	SC000000	LA020000	LK000000	SA060000	ND010000
-2	LA150000	LE190000	LA160000	LE180000	LE010000	LK010000	L5010000	SC090000	LB020000	LK020000	L5020000	ND020000
-3	SM110000	LE170000	SM060000	LE190000	LC010000	LL010000	LT010000	SD030000	LC020000	LL020000	LT020000	ND030000
-4	LA130000	LE130000	LA140000	LE140000	LD010000	LM010000	LU010000	SM050000	LD020000	LM030000	LU020000	ND040000
-5	LA110000	LU100000	LA120000	LU120000	LE010000	LN010000	LV010000	SM050000	LE020000	LN020000	LV020000	ND050000
-6	LA190000	LU150000	LA200000	LU180000	LF010000	LO010000	LP010000	SM250000	LF020000	LO030000	LP020000	ND060000
-7	LA270000	LU170000	LA280000	LU180000	LG010000	LF010000	LX010000	NF040000	LG020000	LF020000	LX020000	ND070000
-8	LC410000	LU130000	LC030000	LU140000	LH010000	LQ010000	LY010000	NF010000	LH020000	LQ030000	LY020000	ND080000
-9	LN190000	SD190000	LN000000	SD130000	LU010000	LR010000	LZ010000	NF060000	LU020000	LR020000	LZ020000	ND090000
-A	LA180000	LU180000	LO170000	SP130000	SP170000	SM210000	SP030000	SM060000	SP030000	ND011000	ND021000	ND031000
-B	SP110000	SC030000	SP030000	SM010000	SP100000	SM200000	SP160000	SM130000	LO150000	LU150000	LO180000	LU160000
-C	SA030000	SM040000	SM020000	SM040000	LD060000	LA010000	LD030000	SM150000	SM050000	SM140000	SM070000	SM080000
-D	SP060000	SP070000	SP090000	SP050000	LY110000	SD410000	LY120000	SD170000	LO130000	LU130000	LO140000	LU140000
-E	SA010000	SP140000	SA050000	SA040000	LT050000	LA020000	LT040000	SD110000	LO110000	LU110000	LO120000	LU120000
-F	SP020000	SD150000	SP150000	SP040000	SA030000	SC010000	SM030000	SA070000	LO190000	LY170000	LO200000	LO000000

Code Page 00273

# Code Page 1025: Cyrillic

HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	ИР 8F010000	& 8M030000	- 8P100000	НЬ KN120000	ц KC010000	й KJ110000	я KA150000	ь KX110000	{ SM110000	}	\ SM070000	0 ND100000
-1	ИРР 8F300000	ЛЬ KL410000	/ 8P120000	Ѣ KC120000	а LA010000	ј LJ010000	~ 8D150000	ы KY010000	А LA020000	Ј LJ020000	§ SM240000	1 ND010000
-2	ћ KD0510000	Ь KN110000	ѓ KG120000	Ќ KK120000	б LB010000	к LK010000	ѕ LB010000	з KZ010000	В LB020000	К LK020000	Ѕ LB020000	2 ND020000
-3	ф KG110000	h KC110000	Ё KE180000	Ѣ 8F320000	с LD010000	l LJ010000	t LT010000	ш KG210000	С LD020000	Л LJ020000	Т LT020000	3 ND030000
-4	ё KE170000	ќ KK110000	Є KE160000	Ў KU240000	d LD010000	m LM010000	u LU010000	э KE130000	Д LD020000	М LM020000	U LU020000	4 ND040000
-5	е KE150000	ђ KU230000	Ѕ KZ160000	Ц KG220000	e LE010000	n LN010000	ч LV010000	ш KG150000	Е LE020000	Н LN020000	В LV020000	5 ND050000
-6	s KZ150000	ц KG210000	І KI120000	ю KU150000	f LF010000	o LO010000	w LW010000	ч KC210000	Ф LF020000	О LO020000	W LW020000	6 ND060000
-7	i KI110000	Ъ KU220000	Ў KI180000	а KA010000	g LG010000	p LP010000	x LX010000	ь KU210000	Г LG020000	Р LP020000	Х LX020000	7 ND070000
-8	ї KI170000	Њ SM030000	Ј KJ020000	б KB010000	h LH010000	q LQ010000	у LY010000	Ю KU160000	Н LH020000	Q LQ020000	У LY020000	8 ND080000
-9	j KJ010000	Ђ KD620000	Љ KL420000	` 8D130000	i LI010000	r LR010000	z LZ010000	А KA020000	І LI020000	Р LR020000	З LZ020000	9 ND090000
-A	[ SM050000	] SM060000	 SM130000	: 8F130000	д KD010000	к KK010000	р KR010000	Б KB020000	Х KH020000	Н KN020000	Т KT020000	З KZ020000
-B	. 8F110000	\$ 8C030000	, 8P080000	# SM010000	e KE010000	л KL010000	с KS010000	Ц KC020000	И KI020000	О KO020000	У KU020000	Ш KS220000
-C	< 8A030000	* 8M040000	% 8M020000	@ 8M050000	ф KF010000	м KM010000	т KT010000	Д KD020000	Й KJ120000	П KP020000	Ж KZ220000	Э KE140000
-D	( 8F060000	) 8F070000	_ 8P050000	' 8F050000	г KG010000	н KN010000	у KU010000	Е KE020000	К KK020000	Я KA160000	В KV020000	Щ KB160000
-E	+ 8A010000	; 8F140000	> 8A050000	= 8A040000	х KH010000	о KO010000	ж KZ210000	Ф KF020000	Л KL020000	Р KR020000	Ь KY120000	Ч KC220000
-F	! 8F020000	^ 8D150000	? 8P150000	" 8F040000	и KI010000	п KP010000	в KV010000	Г KG020000	М KM020000	С KS020000	Ы KY020000	Ѡ 8D010000

# Unicode information

- Example Unicode scripts supported
  - Armenian, Ethiopic, Devanagari, Mongolian, Cherokee, Lao, Deseret, Arabic, Hebrew, Ancient Greek, Musical Symbols, Tibetan and many more.
  - Characters have full names like  
LATIN CAPITAL LETTER A  
or  
BENGALI CURRENCY NUMERATOR ONE LESS THAN THE DENOMINATOR
  - Also use U+xxxx to refer like  
U+0041 or U+0958

# Encoded chars examples

The string “AaÅ” (the character A with Ring accent)

- ASCII
  - x'41', x'61', x'C5'
- EBCDIC
  - x'C1', x'91', x'67'
- Unicode UTF-8
  - x'41', x'61', x'C385'
  - Note: ASCII x'C5' becomes multibyte in UTF-8
- Unicode UTF-16
  - X'0041', x'0061', x'00C5'

# Trademarks and Disclaimers

© IBM Corporation 1994-2007. All rights reserved.

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

Trademarks of International Business Machines Corporation in the United States, other countries, or both can be found on the World Wide Web at <http://www.ibm.com/legal/copytrade.shtml>.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

The customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Some information addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Prices are suggested U.S. list prices and are subject to change without notice. Starting price may not include a hard drive, operating system or other features. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Photographs shown may be engineering prototypes. Changes may be incorporated in production models.