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

Bruce Vining
Session: 510061

i want stress-free IT.
i want control.
i want an i.
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.
Let's 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**

<table>
<thead>
<tr>
<th>Hex Value</th>
<th>4-</th>
<th>5-</th>
<th>6-</th>
<th>7-</th>
<th>8-</th>
<th>9-</th>
<th>A-</th>
<th>B-</th>
<th>C-</th>
<th>D-</th>
<th>E-</th>
<th>F-</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>&amp;</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>/</td>
<td>a</td>
<td>j</td>
<td>A</td>
<td>J</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>02</td>
<td>b</td>
<td>k</td>
<td>s</td>
<td>B</td>
<td>K</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>03</td>
<td>c</td>
<td>l</td>
<td>t</td>
<td>C</td>
<td>L</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>04</td>
<td>d</td>
<td>m</td>
<td>u</td>
<td>D</td>
<td>M</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>05</td>
<td>e</td>
<td>v</td>
<td></td>
<td>E</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>06</td>
<td>f</td>
<td>o</td>
<td>w</td>
<td>F</td>
<td>O</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>07</td>
<td>g</td>
<td>p</td>
<td>x</td>
<td>G</td>
<td>P</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>08</td>
<td>h</td>
<td>q</td>
<td>y</td>
<td>H</td>
<td>Q</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09</td>
<td>i</td>
<td>r</td>
<td>z</td>
<td>I</td>
<td>R</td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0A</td>
<td>%</td>
<td></td>
<td></td>
<td>%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0B</td>
<td>#</td>
<td></td>
<td></td>
<td>#</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0C</td>
<td>@</td>
<td></td>
<td></td>
<td>@</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0D</td>
<td>(</td>
<td></td>
<td></td>
<td>(</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0E</td>
<td>)</td>
<td></td>
<td></td>
<td>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0F</td>
<td>?</td>
<td></td>
<td></td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Fixed Code Points**

Examples of Characters that do change hex values:

#, $, @, Å
Example of a ASCII code page

<table>
<thead>
<tr>
<th>Fixed Code Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changeable Code Points</td>
</tr>
</tbody>
</table>

```
| CODES       | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7- | 8- | 9- | A- | B- | C- | D- | E- | F- |
|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0-          | A  | B  | C  | D  | E  | F  | G  | H  | I  | J  | K  | L  | M  | N  | O  | P  |
| 1-          | Q  | R  | S  | T  | U  | V  | W  | X  | Y  | Z  | a  | b  | c  | d  | e  | f  |
| 2-          | g  | h  | i  | j  | k  | l  | m  | n  | o  | p  | q  | r  | s  | t  | u  | v  |
| 3-          | w  | x  | y  | z  | A  | B  | C  | D  | E  | F  | G  | H  | I  | J  | K  | L  |
| 4-          | M  | N  | O  | P  | Q  | R  | S  | T  | U  | V  | W  | X  | Y  | Z  | a  | b  |
| 5-          | c  | d  | e  | f  | g  | h  | i  | j  | k  | l  | m  | n  | o  | p  | q  | r  |
| 6-          | s  | t  | u  | v  | w  | x  | y  | z  | A  | B  | C  | D  | E  | F  | G  | H  |
| 7-          | I  | J  | K  | L  | M  | N  | O  | P  | Q  | R  | S  | T  | U  | V  | W  | X  |
| 8-          | y  | z  | A  | B  | C  | D  | E  | F  | G  | H  | I  | J  | K  | L  | M  | N  |
| 9-          | O  | P  | Q  | R  | S  | T  | U  | V  | W  | X  | Y  | Z  | a  | b  | c  | d  |
| A-          | e  | f  | g  | h  | i  | j  | k  | l  | m  | n  | o  | p  | q  | r  | s  | t  |
| B-          | u  | v  | w  | x  | y  | z  | A  | B  | C  | D  | E  | F  | G  | H  | I  | J  |
| C-          | k  | l  | m  | n  | o  | p  | q  | r  | s  | t  | u  | v  | w  | x  | y  | z  |
| D-          | o  | p  | q  | r  | s  | t  | u  | v  | w  | x  | y  | z  | A  | B  | C  | D  |
| E-          | t  | u  | v  | w  | x  | y  | z  | A  | B  | C  | D  | E  | F  | G  | H  | I  |
| F-          | z  | A  | B  | C  | D  | E  | F  | G  | H  | I  | J  | K  | L  | M  | N  | O  |
```
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

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

......

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.

<table>
<thead>
<tr>
<th>Country</th>
<th>Keyboard Type</th>
<th>Code page</th>
<th>CCSID</th>
<th>Code point</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>U.S.</td>
<td>USB</td>
<td>037</td>
<td>65535</td>
<td>X’5B’</td>
<td>$</td>
</tr>
<tr>
<td>U.K.</td>
<td>UKB</td>
<td>285</td>
<td>65535</td>
<td>X’5B’</td>
<td>£</td>
</tr>
<tr>
<td>Denmark</td>
<td>DMB</td>
<td>277</td>
<td>65535</td>
<td>X’5B’</td>
<td>Å</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Country</th>
<th>Keyboard Type</th>
<th>Code page</th>
<th>CCSID</th>
<th>Code point</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>U.S.</td>
<td>USB</td>
<td>037</td>
<td>00037</td>
<td>X’5B’</td>
<td>$</td>
</tr>
<tr>
<td>U.K.</td>
<td>UKB</td>
<td>285</td>
<td>00285</td>
<td>X’4A’</td>
<td>$</td>
</tr>
<tr>
<td>Denmark</td>
<td>DMB</td>
<td>277</td>
<td>00277</td>
<td>X’67’</td>
<td>$</td>
</tr>
</tbody>
</table>
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

Order Number . . . . . . 00001

F3=Exit

I902 - Session successfully started
Sample Interactive Ship To Display Using English and CCSID 37

Company: ABC Company
Contact: Bruce Vining

Ship to address: 3605 Highway 52 North
Rochester, MN 55901

<table>
<thead>
<tr>
<th>Part No</th>
<th>Part Description</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hammer</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>Nail</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Wrench</td>
<td>7</td>
</tr>
</tbody>
</table>

F3=Exit
Sample Interactive Ship To Physical File DDS

ORDER (Order Summary):

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

INVEN (Inventory Descriptions):

R INVREC
PARTNO     5  0
PARTDESC   40
K PARTNO

ORDDET (Order Detail):

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
    3  2 'Order Number . . . . . . .' 
      ORDNO   R   I  3 28REFFLD(ORDREC/ORDNO ORDER)
    50
      22  2 'Incorrect Order Number'
      23  4 'F3=Exit'

* Subfile for parts ordered
  R SFLRCD
    SFL
      PARTNO   R   O 12  4REFFLD(ORDDEC/PARTNO ORDDDET)
      PARTDESC  R   O 12 12REFFLD(INVREC/PARTDESC INVEN)
      ORDERQTY  R   O 12 65EDTWRD('

      EDTWRD(' , ')
      REFFLD(ORDDEC/ORDERQTY ORDDDET)
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
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
  write Fmt1
  if RelRecNbr > 0
  eval *in21 = *on
  endif
  exfmt SflCtl
  eval *in21 = *off

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

  eval *inlr = *on
  return
Approach to Inventory Parts Descriptions
Sample Interactive Ship To Display Using German Part Descriptions and CCSID 37

Company: ABC Company
Contact: Bruce Vining
Status: 0
Ship to address: 3605 Highway 52 North
              Rochester, MN 55901

<table>
<thead>
<tr>
<th>Part No</th>
<th>Part Description</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hammer</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>Nagel</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Schraubenschlüssel</td>
<td>7</td>
</tr>
</tbody>
</table>

F3=Exit
Sample Interactive Ship To Display
Using German Part Descriptions and Cyrillic Company Information – Display configured as Cyrillic (CCSID 1025)

<table>
<thead>
<tr>
<th>Part No</th>
<th>Part Description</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hammer</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>Nagel</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Schraubenschl</td>
<td>7</td>
</tr>
</tbody>
</table>

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.
The Answer is Unicode and an Emulator such as System i Access for Web
How about Russian, Chinese, and German?
On the same panel, or different orders on same device at different times

ship to information

Company........... Russian & China world coverage
Contact............. 成福元
Status.............. 1
Ship to address..... Красногородская наб., д. 18
123317 г. Москва, Россия,

Part No | Part Description | Quantity
---------|-----------------|-------
1        | Hammer          | 13    
2        | Negei           | 4     
3        | Schraubenschlussel | 7     

F3=Exit

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

IBM | iSeries | Service

© 2007 IBM Corporation

i want an i.
Only Database Definition Changes to Support Unicode for This Example

- ORDER file:

  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:

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

- ORDER DETAIL file:

  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**

```plaintext
h dftactgrp(*no)

dSetConvert pr 10i 0

d InputCCSID 10i 0 value

d OutputCCSID 10i 0 value


\[ dConvert pr 10i 0 \]

\[ d Input * value \]

\[ d Len_Input 10i 0 value \]

\[ dEndConvert pr 10i 0 extproc('iconv_close') \]

\[ d ConvDesc value like(cd) \]
```

- **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
    if RtnCde = -1
    'Text Error' dspy
    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)

eval RtnCde = Convert(%addr(Input_Variable))

if RtnCde = -1
    'Number Error'dsply
else
    endif
```

* Output_Value now contains the converted field with a length of

* Len_Output bytes
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**

<table>
<thead>
<tr>
<th>pSetConvert</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>dSetConvert</td>
<td>pi</td>
</tr>
<tr>
<td>d InputCCSID</td>
<td>10i 0 value</td>
</tr>
<tr>
<td>d OutputCCSID</td>
<td>10i 0 value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dConvertOpen</th>
<th>pr</th>
<th>52a</th>
<th>extproc('QtqIconvOpen')</th>
</tr>
</thead>
<tbody>
<tr>
<td>d ToCode</td>
<td>* value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d FromCode</td>
<td>* value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dToCode</th>
<th>ds</th>
</tr>
</thead>
<tbody>
<tr>
<td>d ToCCSID</td>
<td>10i 0</td>
</tr>
<tr>
<td>d ToConvAlt</td>
<td>10i 0 inz(0)</td>
</tr>
<tr>
<td>d ToSubAlt</td>
<td>10i 0 inz(0)</td>
</tr>
<tr>
<td>d ToStateAlt</td>
<td>10i 0 inz(0)</td>
</tr>
<tr>
<td>d ToLenOpt</td>
<td>10i 0 inz(0)</td>
</tr>
<tr>
<td>d ToErrOpt</td>
<td>10i 0 inz(0)</td>
</tr>
<tr>
<td>d TReserved</td>
<td>8 inz(*allx'00')</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dFromCode</th>
<th>ds</th>
</tr>
</thead>
<tbody>
<tr>
<td>d FromCCSID</td>
<td>10i 0</td>
</tr>
<tr>
<td>d FromConvAlt</td>
<td>10i 0 inz(0)</td>
</tr>
<tr>
<td>d FromSubAlt</td>
<td>10i 0 inz(0)</td>
</tr>
<tr>
<td>d FromStateAlt</td>
<td>10i 0 inz(0)</td>
</tr>
<tr>
<td>d FromLenOpt</td>
<td>10i 0 inz(0)</td>
</tr>
<tr>
<td>d FromErrOpt</td>
<td>10i 0 inz(0)</td>
</tr>
<tr>
<td>d FReserved</td>
<td>8 inz(*allx'00')</td>
</tr>
</tbody>
</table>
iconv
SetConvert common routine

```c
  eval  FromCCSID = InputCCSID
  eval  ToCCSID = OutputCCSID
  eval  cd = ConvertOpen( %addr(ToCode)
                        :%addr(FromCode))
  if        cdBins(1) = -1
    'Open Error'  dsply
  return    -1
else
  return    0
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
               :%addr(InputPointer)
               :InBytesLeft
               :%addr(OutBufPtr)
               :OutBytesLeft)
c 'Conv Error' dsply
  return -1
c else
  eval Len_Output = %len(Output_Value) - OutBytesLeft
  return 0
  endif
pConvert
References

• System i globalization home page
  – http://www-03.ibm.com/servers/eserver/iseries/software/globalization/

• List & view of IBM codepages

• G11N Api’s

• 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-ORDDET
      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.
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 ORDDET.
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 SFLCTRL-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
  END-READ

  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 Sf1Ctl-0 OF Sf1Ctl-0-DS
  MOVE 0 TO RelRecNbr
  MOVE B"1" TO IN25 OF Sf1Ctl-0-DS
  WRITE Ship-To-DSPF-Records FROM Sf1Ctl-0 OF Sf1Ctl-0-DS FORMAT IS "SFLCTL"
  MOVE B"0" TO IN25 OF Sf1Ctl-0-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 Sf1Rcd-0 OF Sf1Rcd-0-DS
    MOVE CORR InvRec TO Sf1Rcd-0 OF Sf1Rcd-0-DS
    WRITE SUBFILE Ship-To-DSPF-Records FROM Sf1Rcd-0-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.
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.
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(&OUTBYTELEFT) 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) DEFFAR(&CD)

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

DCL VAR(&TOCODE) TYPE(*CHAR) LEN(32)
DCL VAR(&TCCCSID) TYPE(*INT) STG(*DEFINED) +
DEFVAR(&TOCODE)
iconv - CL

/* Initialize &TOCODE and &FROMCODE to all x'00' */
/* 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)
SNDPGMM 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(&OUTBYTELEFT) VALUE(4096)
CALLPRC PRC('iconv') PARM((&CD *BYVAL) (&INPUTPTR) +
   (&INBYTELEFT) (&OUTPUTPTR) (&OUTBYTELEFT)) +
   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(&OUTBYTELEFT) VALUE(4096)
CALLPRC PRC('iconv') PARM((&CD *BYVAL) (&INPUTPTR) +
               (&INBYTELEFT) (&OUTPUTPTR) (&OUTBYTELEFT)) +
               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

<table>
<thead>
<tr>
<th>HEX DIGITS</th>
<th>4-</th>
<th>5-</th>
<th>6-</th>
<th>7-</th>
<th>8-</th>
<th>9-</th>
<th>A-</th>
<th>B-</th>
<th>C-</th>
<th>D-</th>
<th>E-</th>
<th>F-</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>-1</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>-2</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>-3</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>-4</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>-5</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>-6</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>-7</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>-8</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>-9</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>-A</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>-B</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>-C</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>-D</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>-E</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>-F</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>A0</td>
<td>C0</td>
<td>E0</td>
<td>00</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
</tbody>
</table>

Code Page 00037
### Code Page 273: German

<table>
<thead>
<tr>
<th>HEX</th>
<th>4-</th>
<th>5-</th>
<th>6-</th>
<th>7-</th>
<th>8-</th>
<th>9-</th>
<th>A-</th>
<th>B-</th>
<th>C-</th>
<th>D-</th>
<th>E-</th>
<th>F-</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>⊗</td>
<td>&amp;</td>
<td>-</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ø</td>
<td>ö</td>
<td>ö</td>
<td>ö</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
</tr>
<tr>
<td>2</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
</tr>
<tr>
<td>3</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
<td>ñ</td>
</tr>
<tr>
<td>4</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
</tr>
<tr>
<td>5</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
</tr>
<tr>
<td>6</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
</tr>
<tr>
<td>7</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
</tr>
<tr>
<td>8</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
</tr>
<tr>
<td>9</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
</tr>
<tr>
<td>10</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
</tr>
<tr>
<td>11</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
</tr>
<tr>
<td>12</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
</tr>
<tr>
<td>13</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
</tr>
<tr>
<td>14</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
</tr>
<tr>
<td>15</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
<td>å</td>
</tr>
</tbody>
</table>

*Code Page 00273*
### Code Page 1025: Cyrillic

<table>
<thead>
<tr>
<th>DEC-DIGITS</th>
<th>1ST</th>
<th>2ND</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2F</td>
<td>й</td>
<td>33</td>
<td>ё</td>
<td>37</td>
<td>ї</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2F</td>
<td>й</td>
<td>33</td>
<td>ё</td>
<td>37</td>
<td>ї</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2F</td>
<td>й</td>
<td>33</td>
<td>ё</td>
<td>37</td>
<td>ї</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2F</td>
<td>й</td>
<td>33</td>
<td>ё</td>
<td>37</td>
<td>ї</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2F</td>
<td>й</td>
<td>33</td>
<td>ё</td>
<td>37</td>
<td>ї</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2F</td>
<td>й</td>
<td>33</td>
<td>ё</td>
<td>37</td>
<td>ї</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2F</td>
<td>й</td>
<td>33</td>
<td>ё</td>
<td>37</td>
<td>ї</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2F</td>
<td>й</td>
<td>33</td>
<td>ё</td>
<td>37</td>
<td>ї</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### i want an i.

© 2007 IBM Corporation
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'61x, 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.


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.