IBM

DBCS Design Guide - System/370 Software

GG18-9095-0

IBM

DBCS Design Guide - System/370 Software

GG18-9095-0

First Edition (October 1988)

Changes are made occasionally to the information herein; any such changes will be reported in subsequent revisions.

Preface

The objective of this Design Guide is to provide developers of S/370 application software with information on what is required to write programs that are capable to handle DBCS characters such as Chinese, Japanese, or Korean, and to translate messages and panels issued by application programs into such languages.

The term DBCS (Double-Byte Character Set) is used to describe a graphic character set where a character is represented by two bytes. The term SBCS (Single-Byte Character Set) is used to refer to a character set where a character is represented by one byte.

Our focused "application software" throughout this manual is software for a specific application such as personnel application or accounting application which has the following characteristics:

- Transaction processing oriented program
- Access to data base designed in the application

Application unique requirements other than DBCS such as language unique functions are not discussed in this manual. Also, you should verify that laws, business customs, etc., will permit your application to run in the target country, before investing in modifications of software for DBCS.

Who should read this manual?

This manual is written for a broad audience who have an interest in DBCS, including planners, designers, and programmers of application software. It is assumed that the audience is familiar with the planning, designing, and developing application software under the SBCS environment.

- For planners, this manual explains the environment surrounding DBCS applications from the user requirements, the development process, and other points of view.
- For designers, this manual shows the total picture for the DBCS application; what are required when applying your applications to Asian countries, what tools or utilities are available to improve the productivity when writing DBCS applications, and the key considerations needed when translating messages in your applications into different national languages.
- For programmers, this manual provides basic information on DBCS application; the specific functions provided for handling DBCS characters, the device data stream for DBCS support, and other specific considerations.

The table below is a topical guide for each audience group's interests and concerns. This table may be useful to you in deciding what to read.

Section	**Planner**	**Designer**	**Programmer**
Chapter 1, "Introduction"	X	X	X
Chapter 2, "DBCS System"	X	X	X
Chapter 3, "Processing DBCS"		X	X
Chapter 4, "Presenting DBCS"		X	X
Chapter 5, "DBCS Enabled IBM Key Software"		X	X
Chapter 6, "Message Translation Prerequisites"		X	X
Appendix A, "Device Data Stream"			X
Appendix B, "Advanced Presentation"		X	X
Appendix C, "Common DBCS Processing Functions"		X	X
Appendix D, "DBCS Testing"	X	X	X
Appendix E, "PS/55"	X	X	X
Appendix F, "DBCS-PC"		X	X
Appendix G, "Product List"	X	X	X

Figure 1. Topical Guide of DBCS Design Guide

Reference manuals

The following publications provide general and specific information on DBCS and National Language Support (NLS).

- National Language Information and Design Guide
  Volume 1: Designing Enabled Products, Rules/Guideline, SE09-8001
  Volume 2: Left-to-Right Languages and DBCS Languages, SE09-8002
- S/370 - DBCS Applications Primer
  (Enabling your programs for Chinese/Japanese/Korean), GG18-9059
- SAA Common User Access:
  Panel Design and User Instruction, SC26-4358
- IBM 3270 Information Display System Data Stream:
  Programmer's Reference, GA23-0059

Contents

Figures

Notations

The following notations are used in this document.

DBCS CHARACTER REPRESENTATION

DBCS characters in examples are represented by:

- NORMAL PRESENTATION

```
DiDjDkDl
D B C S
```

  where ..,i,j,k,.. indicates a position in a data stream or in a presentation space.

- HEXADECIMAL REPRESENTATION

  X'xxyy' is the general representation of a DBCS character where xx and yy are in the range shown in the DBCS code table. (Refer to Figure 2 on page 4.)

SPECIFIC DBCS CHARACTER REPRESENTATION

- DOUBLE-BYTE ALPHANUMERIC / SPECIAL SYMBOLS

  DBCS specific characters representing double-byte alphanumeric and special symbols are represented as follows:

```
.A.B.C
A B C
```

  Both of the above represents double-byte A, B, C.

- SPECIFIC CHARACTER HEXADECIMAL REPRESENTATION

  X'nnyy' or X'yynn' where nn is shown specifically (e.g. nn=x'42') refers to specific DBCS characters with that value as their first or second byte. For example x'42yy' is any specific DBCS character which begins with X'42'.

SBCS CHARACTER REPRESENTATION

SBCS (EBCDIC) characters in examples are represented by:

- NORMAL PRESENTATION

```
sssssss
abcd...
```

- HEXADECIMAL REPRESENTATION

  X'xx' is the hexadecimal representation of a SBCS character.

SHIFT IN AND SHIFT OUT (see page 1 for explanation of SO/SI.)

The following is used for SO/SI in examples:

```
␎  : SO        ␏ : SI

sss␎D1D2D3D4␏ss
sss␎ＤＢＣＳ␏ss
```

1.0 Introduction

WELCOME TO A NEW DATA WORLD!

The growth of computers has affected users all over the world. A user's requirement to use a computer in a "national language" is very rational, especially for end-users. In countries where ideographic characters such as Chinese, Japanese and Korean are used, it is a fundamental requirement to handle DBCS (Double-Byte Character Set) as well as SBCS (Single-Byte Character Set) in the application software to support this "Ease of Use" movement. The objective of this document is to explain DBCS support so that you will have a clear understanding if it.

What is DBCS?: Some languages have too many symbols to be able to represent all the characters using one-byte codes (SBCS). For example, Kanji used in the Japanese language, Hanzi used in the Chinese language, and Hangeul used in the Korean language contain several thousands characters. To create coded character sets for such languages, we need two bytes for each character. Let's consider what concerns may arise by introducing DBCS.

Is knowledge of the language itself necessary? The development of DBCS applications falls into two steps; function development step and translation step. As far as the function development step is concerned, you need no language knowledge[1] because DBCS contains the alphanumeric characters familiar to the readers who use the Latin characters. This means that you are able to do testing of DBCS functions using the DBCS alphanumerics. The remainder of the DBCS may be regarded just as symbols, as if the user wanted to handle mathematical expressions such as $\sum$, $\int$. However, translators need a specific language knowledge during the translation step where messages and panels are translated into the national language.

What is DBCS data?: DBCS data takes two forms. One is a string consisting only of Double-Byte Characters, called a DBCS string. All operations on a DBCS string will assume that two bytes are used for each character. The other is a string made of a mixture of SBCS and DBCS, called a Mixed string. The DBCS portions in a Mixed string are bracketed by delimiters to identify DBCS characters from SBCS. Shift-out (SO) indicates the beginning of DBCS and Shift-in (SI) indicates the end of DBCS.[2]

```
Example)
        DBCS string      D D D D
        Mixed string     ssss␎D D D D␏
```

[1] You need the specific language knowledge if your product requires linguistics functions such as text processing.

[2] The code points of SO/SI characters are X'0E' and X'0F' respectively.

What DBCS I/O devices are available?: The DBCS workstations of IBM PS/55TM [3] provide various display fields which accept and display DBCS strings and Mixed strings. The IBM PS/55 terminal printers also have a capability to print DBCS data. DBCS print data generated by application software can be printed on other printers such as the IBM 3800-8 and IBM 3820.

What is the DBCS application environment?: Most of the presentation services provide a programming interface to the DBCS workstation and the terminal printers. DBCS print data generated by application software both in the line print format and in the page format are processed by some printer service products to be converted to the data stream for the target printer. Various functions on DBCS or Mixed strings are also available in some programming languages and data base managers.

What is in this manual?: The application software assumed in this publication is a typical transaction process oriented program. This manual introduces you to detailed design points to support DBCS.

Basic information about DBCS important to the designers and/or the programmers is described in the following chapters.

- Chapter 2.0, "DBCS System"
- Chapter 3.0, "Processing DBCS"
- Chapter 4.0, "Presenting DBCS"

The following chapter gives information on the individual IBM products that provide DBCS functions to application programs.

- Chapter 5.0, "DBCS Enabled IBM Key Software"

Additional consideration is required to provide MRI (Machine Readable Information)[4] translation into national languages.

- Chapter 6.0, "Message Translation Prerequisites"

The appendixes contain information on other items related to DBCS support.

- Appendix A, "Device Data Stream"
- Appendix B, "Advanced Presentation"
- Appendix C, "Common DBCS Processing Functions"
- Appendix D, "DBCS Testing"
- Appendix E, "PS/55"
- Appendix F, "DBCS-PC"

[3] IBM PS/55 is a trademark of the IBM Corporation.

[4] Machine Readable Information means all textual information contained in a program, which may appear on display panels or printers.

2.0 DBCS System

This section provides a basic knowledge of the DBCS system environment. In the first two sections, you will see the structure of the DBCS followed by the DBCS unique considerations. The third and fourth sections give you an overview of the DBCS system environment from the viewpoint of application software development.

2.1 Character Sets

IBM defines various Double-Byte Character Sets for the Asian countries. DBCS workstations such as the IBM PS/55 are designed to handle two character sets at a time, one from Double-byte Character Sets and the other from Single-byte Character Sets. DBCS system printers such as the IBM 3800-8 and IBM 3820 also support multiple character sets. For the applications used in the Asian countries, it is required to support the dual character sets; Double- and Single-Byte Character Set.

2.1.1 DBCS

A Double-Byte Character Set (DBCS)[5] is a graphic character set in which each character is represented by two bytes. DBCS codes meet the following criteria (common among all the DBCSs). DBCS does not have any control characters.[6]

- X'4040' represents a DBCS blank.
- First byte is in the range X'41' to 'X'FE'.
- Second byte is in the range X'41' to X'FE'.
- All other undefined 16-bits patterns are invalid.

[5] There are two types of DBCS defined in IBM system environments; DBCS-HOST for S/370 and S/3X environment and DBCS-PC for PC environment. DBCS-PC code based on ASCII is different from DBCS-HOST code based on EBCDIC. (Refer to Appendix F, "DBCS-PC" on page 113.) Throughout this manual, the term DBCS refers to DBCS-HOST.

[6] Some DBCS control characters are defined for printers. However, those characters are device dependent, and are used by printer device support programs and are generally not needed by application programs.

X'0000' can be treated in the same manner as a program does X'00'. In this case, X'0000' should not be rejected as an invalid code but suppressed as a DBCS Null.

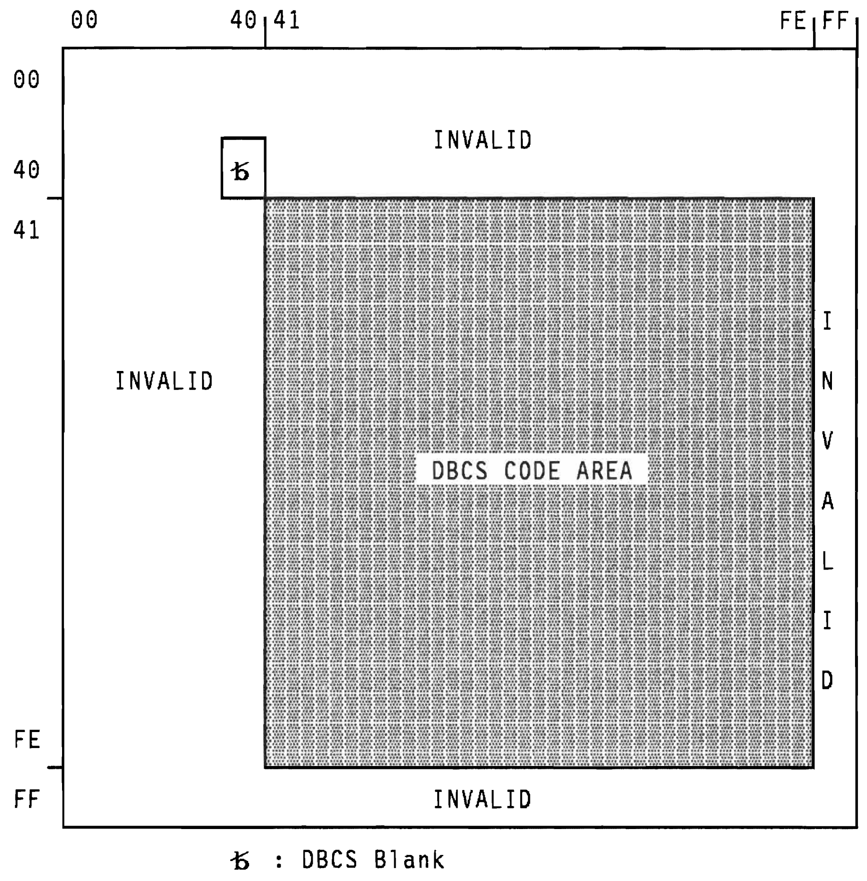


Figure 2. DBCS Code Table

42nd ward DBCS character: All DBCSs, Chinese, Japanese and Korean, contain common double-byte alphanumeric and special symbols that correspond to those of SBCS. The first byte of the double-byte alphanumeric/special symbols is X′42′ and the second byte is the hexadecimal value of a corresponding EBCDIC code. This portion of DBCS codes is called “42nd ward DBCS character” .

42 WARD

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
424											☆	.	<	(	+	|
425	&										!	☆	*	)	;	¬
426	−	/									¦	,	%	_	>	?
427											:	#	@	'	=	"
428		a	b	c	d	e	f	g	h	i						
429		j	k	l	m	n	o	p	q	r						
42A		☆	s	t	u	v	w	x	y	z						
42B																
42C	{	A	B	C	D	E	F	G	H	I						
42D	}	J	K	L	M	N	O	P	Q	R						
42E	☆		S	T	U	V	W	X	Y	Z						
42F	0	1	2	3	4	5	6	7	8	9						

Note : The four code points indicated by ☆ above are reserved for National Use graphic characters.

2.1.2 SBCS

SBCS (Single-Byte Character Set) is a character set where each character falls within the range of X'00' to X'FF'. These characters have been traditionally labeled EBCDIC.

	SBCS	DBCS
Control characters	X'00'-X'3F'	none
Control	X'FF'	none
Blank	X'40'	X'4040'
Graphics	X'41'-X'FE'	First byte : X'41'-X'FE' Second byte : X'41'-X'FE'

Figure 3. Comparison of SBCS and DBCS

Japanese-Katakana: There are two choices of SBCS which can be set up for a DBCS workstation used in Japan, U.S-English and Japanese-Katakana. Workstations set up with Japanese-Katakana cannot display lower case single-byte alphabetic character because the same code is assigned to the Katakana character. Programs which use the lower case alphabetic characters should consider this to avoid problems when both U.S-English and Japanese-Katakana terminals are supported. Refer to 6.4, "Katakana Terminal Support" on page 69.

2.2 Strings

There are two types of strings that can contain DBCS characters. They are a DBCS string and a Mixed string. DBCS data sometimes refers to either DBCS or Mixed string in this manual.

```
           ┌ DBCS  string      D1D2D3
DBCS DATA ─┤
           └ Mixed string      sss␎D1D2D3␏ss
```

2.2.1 DBCS String

A DBCS string is a character string that consists of DBCS codes only. Shift-Out (SO) and Shift-In (SI) characters are not associated with DBCS strings. A DBCS string should be manipulated on the basis of the DBCS character boundary, which means operations on a DBCS string should work on a two-byte unit.

```
D1D2D3D4D5 (DBCS string)
D1
| D2
| | D3
| | | D4
| | | | D5
| | | | |
1 2 3 4 5  ----- DBCS Character positions
```

Figure 4. DBCS String

2.2.2 SBCS/DBCS Mixed String

A SBCS/DBCS mixed string is a character string that consists of single-byte (SBCS) portions and double-byte (DBCS) portions. DBCS portions are enclosed with a Shift-Out (SO) character and a Shift-In character (SI) to identify DBCS characters from SBCS characters. The terms SBCS/DBCS mixed strings and Mixed strings are synonymous in this document.

```
          11
12345678901 ----- Byte positions
|||||||||||
||||||||||s
|||||||||s
||||||||▨
|||||||2
||||||D
|||||1
||||D
|||▨
||s
|s
s
sss▨D1D2▨ss  (Mixed string)
s
|s
||s
||| D1
||| |  D2
||| |  |   s
||| |  |   |s
||| |  |   ||
123 4  5   67 ----- Logical Character positions
```

Figure 5. Mixed String

2.3 DBCS I/O Devices

The majority of S/370 hardware components are usable without any modification in configuring a DBCS system. The exception is I/O devices; workstations and printers.

2.3.1 Workstation

The IBM PS/55[7] is a multi-purpose workstation which provides DBCS input/output capability as a 3270 information display, as a personal computer, and as a word processor. It is comparable to the IBM PS/2™ [8] in DBCS countries.

[7] The IBM PS/55 was formerly called IBM 5550. The IBM PS/55 is compatible with the IBM 5550 family.

[8] IBM PS/2 is a registered trademark of the IBM corporation.

The IBM PS/55 with the 3270 PC or 3270 PC/G control program that provides the DBCS character font sets for the individual Asian countries serves as a 3270 terminal. Two control programs, 3270 PC and 3270 PC/G, are selectable at installation time. The control programs provide the common application interface and user interface with those provided under the SBCS environment.

Display

The displays of the IBM PS/55 provide the capability to define the following fields in addition to those that can be defined on SBCS displays. An outlined field (field with field outlining) can be also defined on the display of the IBM PS/55.

DBCS field: A DBCS string is displayed/accepted by a field called a "DBCS field" . There is no SO/SI associated with this field type.

Mixed field: A Mixed string is displayed/accepted by a field called a "Mixed field" .[9]

Keyboard

The keyboard for Asian countries has the unique function keys to input DBCS characters such as the status switching key that is used to change SBCS status to DBCS status. When a DBCS character is keyed in from the keyboard on a Mixed field, a pair of SO/SI characters automatically appear.

Terminal printer

The terminal printers of the IBM PS/55 are available to print DBCS characters in addition to SBCS characters.

2.3.2 System Printer

The IBM 3800-8, IBM 3820, IBM 3827, and IBM 3835 are available to print DBCS character in addition to SBCS characters. To print out data sets containing DBCS characters, PSF (Print Services Facility), a print support program, is required.

[9] A field defined on a DBCS workstation as an ordinary "EBCDIC field" can display a Mixed string sent in an outbound stream.

2.4 DBCS Application Environment

IBM offers various software products that provide application developers with the capability for DBCS presenting and programming. Basically, you do not need to change your current application development and execution environment for DBCS applications because software products for your application environment are enabled to handle DBCS data.
In this section, a brief introduction to DBCS application environment that consists of such software products is described. Additional information is described in 5.0, "DBCS Enabled IBM Key Software" on page 37.

2.4.1 Presentation Support

Display/Terminal printer

Full screen interface: In building a full screen panel under some presentation services such as GDDM, a programming interface is provided for users to access DBCS characters from/to DBCS fields and Mixed fields. By using the same programming interface, DBCS characters can be printed on a terminal printer

Line mode interface: Under environments such as TSO and CMS, an application can pass to/from a display and send to a printer a Mixed string by using line mode.

System printer

The print support programs for the system printers such as PSF (Print Services Facility) support a variety of record formats of the print data that is generated by an application software.

Line print format: In line mode printing, DBCS characters enclosed with SO/SI can be put in the text lines. It is optional whether or not SO/SI characters take print positions (printed as spaces[10]).

Page print format: All functions are DBCS enabled. Text control "Set Coded Font Local" can be used to switch not only SBCS fonts but also DBCS fonts.

[10] It is possible to print SO/SI characters as other characters instead of spaces.

2.4.2 Manipulation Support

Some programming languages and data base managers have programming capability for DBCS data manipulations.

Programming language

To manipulate a DBCS string, "DBCS data type" is provided by some programming languages and data base managers. And a Mixed string can be stored in "Character data type" .

DBCS data type (DBCS data item): Some programming languages keep track of the characteristics of data items through data types such as integer, character and so on. "DBCS data type" is implemented in programming languages such as OS PL/I and VS COBOL II to keep track of the characteristics of the DBCS string. It is called "GRAPHIC" in several products.

Character (SBCS) data type (Character data item): Every programming language supports a data item for SBCS string manipulation. It is implemented either as an explicit data type, which is usually called "CHARACTER," or as an implicit one. A Mixed string can be stored as such a data item regardless of DBCS manipulation capability. In cases when the data item has no DBCS manipulation capability, the manipulation on the Mixed string works on the basis of the byte unit, ignoring the DBCS character boundaries with no special handling for SO/SI. However, there is no problem with the move operation unless it causes truncation of a DBCS substring portion.

A Mixed string can be well-handled in a data field which is implemented with the concept of the logical character. Then a Mixed string is manipulated on the basis of the logical character unit. The logical character support is based on the logical character positioning (refer to Figure 5 on page 8). SO/SI characters are not objects of manipulation, because they are just control characters.

```
Example:

   A = '␎D1D2D3␏sssss'
   B = SUBSTR(A,1,8)

      Under normal Mixed support    B : ␎D1D2D3␏         8 byte
      Under logical char support    B : ␎D1D2D3␏sssss    8 characters
```

Data base manager

Data base managers such as DB2 and SQL/DS provide full support for the DBCS string. The data types for DBCS strings are implemented as "GRAPHIC," "VARGRAPHIC," and "LONG VARGRAPHIC" data type. The associated operations on DBCS strings such as substring and assignment operation are also provided with the DBCS capability. The Mixed string support is similar to that described above in the Character data type of the programming language.

3.0 Processing DBCS

In any operation on DBCS data, the DBCS character boundaries should be recognized properly. This means, each DBCS character should always be handled in a two-byte unit. In addition, the proper pairing of SO/SI character should be assured for the Mixed string.

The DBCS data type gives a convenient approach for the application software which processes DBCS strings. As for Mixed string handling, it is possible to store a Mixed string as an SBCS data item with some restriction on the manipulation.

In the following section you will see the information on these data types supported by the programming languages and data base managers and how to handle Mixed strings.

3.1 DBCS String Handling

3.1.1 DBCS Data Type

It is very straightforward to understand the DBCS data type (implemented as "GRAPHIC data type" in some products). Like the Character (SBCS) data type, there is a close relationship of the programming interface to the DBCS data type among programming languages, data base managers, and presentation service products. In developing DBCS applications, it is recommended to use these programming interfaces.

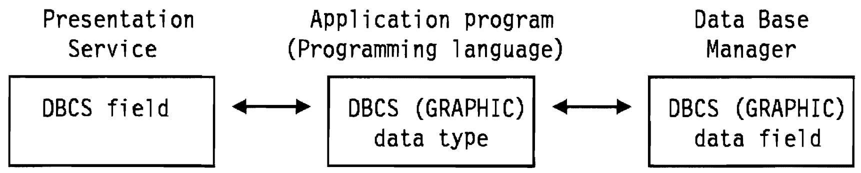


Figure 6. Programming Interface to DBCS Data Type

DBCS literal

A standard format is used to specify a DBCS string as a literal/constant in a source code. A DBCS literal is expressed with a designator, for example "G," and DBCS string bracketed with SO/SI characters. The user's specification format is, for example,

```
G'␎D1D2D3D4␏'
```

This token is processed by the parser and recognized as

```
D1D2D3D4
```

Programming language: Most of the programming languages, such as some high level languages and command interpreters that allow users to declare a data type, provide a programming capability for the DBCS string, a DBCS data type.

Data base manager: DB2 and SQL/DS are relational data base managers which support a data field for the DBCS string. The data base language called SQL provides an programming interface to the DBCS data field.

Presentation: The following products provide programming interfaces for DBCS fields in full screen mode.

- ISPF/DM[11]
- GDDM
- IMS/MFS
- CICS/BMS

See 4.0, "Presenting DBCS" on page 29 for details on how to use them.

3.1.2 Operation for DBCS String

A DBCS string is manipulated in the same manner as an SBCS string except that the operation unit is two-bytes. Programming languages and data base managers that support DBCS data type provide operations for DBCS data items, which have the same syntax as those for SBCS data items.

Truncation

Truncation occurs when a string is moved into a shorter area. It is assured that a DBCS string is always truncated at the position of DBCS character boundaries as far as the DBCS string is stored in a DBCS (GRAPHIC) data item.

[11] The VSE version of ISPF does not provide the interface to DBCS fields.

Padding

When a DBCS string is padded, DBCS blanks are usually used as the padding character. Sometimes a program allows the user to specify the padding character. If a padding character is specified by an SBCS character, that SBCS character must be converted to the corresponding double-byte code in the 42nd ward DBCS characters. (Refer to "42nd ward DBCS character" on page 5.)

3.2 Mixed String Handling

3.2.1 Character (SBCS) Data Type

A Mixed string can be stored as a Character data item in most products, which do not put a restriction on the code point range. If manipulation such as truncation, substring, and search operate on a Mixed string, however, they do not ensure the validity of the result unless it is a simple move operation without truncation. A programmer has to be careful about manipulation which may cause a loss of data contents.

Some products allow Mixed string literals. The format for the Mixed literal is usually the same as that for an SBCS string.[12]

The following shows the relation of the programming interface to the Character data type among programming languages, data base managers, and presentation service products. Note that Mixed string can be displayed on both Mixed and EBCDIC fields defined on IBM PS/55 displays, however, it cannot be keyed-in on EBCDIC fields.

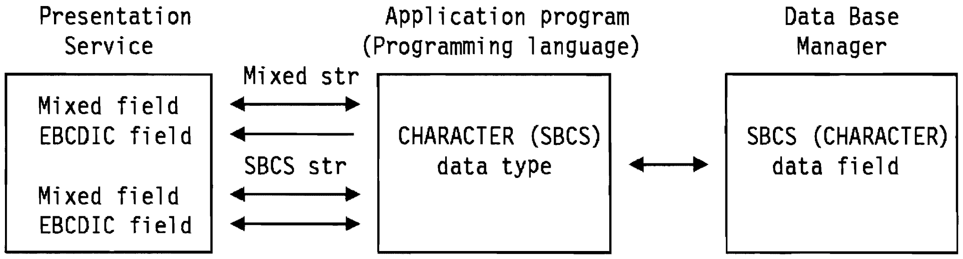


Figure 7. Programming Interface to Character Data Type

[12] In some programs, a Mixed string is intended to be fixed text, which does not need character-based manipulation. With this assumption, Mixed literal support is very useful even if Mixed string manipulations are not provided.

Mixed literal

A DBCS character in a Mixed string specified in a source code may contain the same bit patterns as a special symbol such as a quotation or a slash in its first or second byte. Mixed literals are used to recognize Mixed strings, ignoring such bit patterns in DBCS portions. Generally, Mixed literals are syntactically expressed as character literals.[13]

```
'ss␎D1D2D3␏s'
```

Programming language: It is possible to store a Mixed string as a Character (SBCS) data item with the following restrictions on the manipulations. The move operation works correctly on the Mixed string as if it were an SBCS string unless truncation occurs. If an assignment to a shorter space results in truncation of a Mixed string, it sometimes results in an invalid Mixed string.

Data base manager: DB2 and SQL/DS are relational data base managers that support a data field for the character string, which can be used to store a Mixed string. The data base language SQL provides an programming interface to them.

Presentation: The following products provide programming interfaces for Mixed fields in full screen mode.

- ISPF/DM[14]
- GDDM
- IMS/MFS
- CICS/BMS

See 4.0, "Presenting DBCS" on page 29 for detail on how to use them.

3.2.2 Operation for Mixed String

In applications which treat a Mixed string, it is sometimes required to manipulate a Mixed string while keeping the validity of the Mixed string. In this case, the application should have coding of some special manipulations. In this section you see the information on the typical operations on Mixed strings, which are used instead of those for SBCS strings. Note that all the functions are not necessarily implemented in IBM products. So you have to know the support level of DBCS functions in IBM

[13] A Mixed literal is expressed with a designator, for example "M," in some products.

[14] The VSE version of ISPF does not provide the interface to DBCS fields.

base products such as COBOL, SQL, and GDDM before coding an application (refer to 5.0, "DBCS Enabled IBM Key Software" on page 37). And the application should have coding of some unsupported manipulations by itself, if necessary.

Truncation

When a Mixed string is truncated during a process, the program should fix the result to make it a valid string. It is necessary to add this compensation logic to the manipulations such as assignment, etc.

The substring is a combination of right truncation and left truncation. See Figure 8 and Figure 9 on page 18 for examples on how the Mixed string should be processed.

```
RIGHT TRUNCATION

   Truncated Here      Intermediate      Adjusted
           |           Result            Result
           ▼

 s␎D1D2␏ss             s␎D1D2␏ss         s␎D1D2␏ss
  s␎D1D2␏s s            s␎D1D2␏s          s␎D1D2␏s
   s␎D1D2␏ ss            s␎D1D2␏           s␎D1D2␏
    s␎D1D2 ␏ss            s␎D1D2            s␎D1␏#
     s␎D1D 2␏ss            s␎D1D             s␎D1␏
      s␎D1 D2␏ss            s␎D1              s###
       s␎D 1D2␏ss            s␎D               s##
        s␎ D1D2␏ss            s␎                s#
         s ␎D1D2␏ss            s                 s

                                         #=padding character
                                           such as a blank.


LEFT TRUNCATION

  Truncated Here    Intermediate        Adjusted
          |         Result              Result
          ▼

 s␎D1D2␏s s             s                 s
  s␎D1D2␏ ss            ss                ss
   s␎D1D2 ␏ss           ␏ss               #ss
    s␎D1D 2␏ss          2␏ss              ##ss
     s␎D1 D2␏ss         D2␏ss             ###ss
      s␎D 1D2␏ss        1D2␏ss            ␎D2␏ss
       s␎ D1D2␏ss       D1D2␏ss           #␎D2␏ss
        s ␎D1D2␏ss      ␎D1D2␏ss          ␎D1D2␏ss
          s␎D1D2␏ss     s␎D1D2␏ss         s␎D1D2␏ss

                                      #=padding character
                                        such as a blank.
```

Figure 8. Examples of Compensation of Truncated Mixed String

```
Start positions         End positions
↓  ↓↓  ↓                ↓  ↓↓  ↓

A▒D1D2D3▒BC...          ...A▒D1D2D3▒BC

A▒D1D2D3▒BC...          ...Ab̸
     b̸▒D3▒BC...          ...A▒D1▒
       ▒D3▒BC...          ...A▒D1▒b̸
           b̸BC...          ...A▒D1D2D3▒

                        b̸ = single-byte blank X'40'
```

Figure 9. Examples of Substring Adjustment

Padding

For the languages and the data base managers which maintain the length of the data contents, data is sometimes padded with blanks. Mixed strings are also padded by a SBCS blanks (X′40′).

Search

In order to support a Mixed string for search operation, some changes of logic are required so that comparison should work on a character basis, not a byte basis. A DBCS character is to be compared with a DBCS character in the other string. This means, when a search argument is a DBCS character, any two contiguous SBCS characters should not be searched even if the SBCS characters have the same bit patterns as the DBCS character.

```
Example

  X'4E6E'   in SBCS             =  +>
            in DBCS (Japanese)  =  蝶 (character that means 'butterfly')
```

Similarly, the combination of the second byte in a DBCS character and the first byte in the next DBCS character should not be searched.

In an actual implementation, a search argument is given from a user through a command line where Mixed strings can be entered. This means that the search argument entered by a user is always given in the form of a Mixed string even if a user wants to search a DBCS string. That is, the program needs to adjust a user input string to a proper search argument. SO/SI characters are not significant unless they reside between a SBCS character and a DBCS character.

```
Example 1   SO, SI have no meaning

     user input     search argument    target           result

      ␎Di␏               Di            ␎Di␏             match
                                       ␎--Di--␏         match
                                       ␎----Di␏         match
                                       ␎Di----␏         match


Example 2   SI is significant but SO is not significant

     user input     search argument    target           result

      ␎Di␏ss             Di␏ss         ␎Di␏ss           match
                                       ␎Di----␏ss       no match
                                       ␎--Di--␏ss       no match
                                       ␎----Di␏ss       match
                                       ␎Di␏tss          no match
```

Figure 10. Examples of Search of Mixed String

Adjacent SO/SI Pair

Adjacent SO/SI pairs are often generated after concatenating two Mixed strings. It depends on the applications whether or not to remove adjacent SO/SI pairs. If an application does not care about the length and respects the content only, remove them. If an application cares about the length as well, do not remove the SO/SI pairs. Some applications may replace the SO/SI with the DBCS blank.[15]

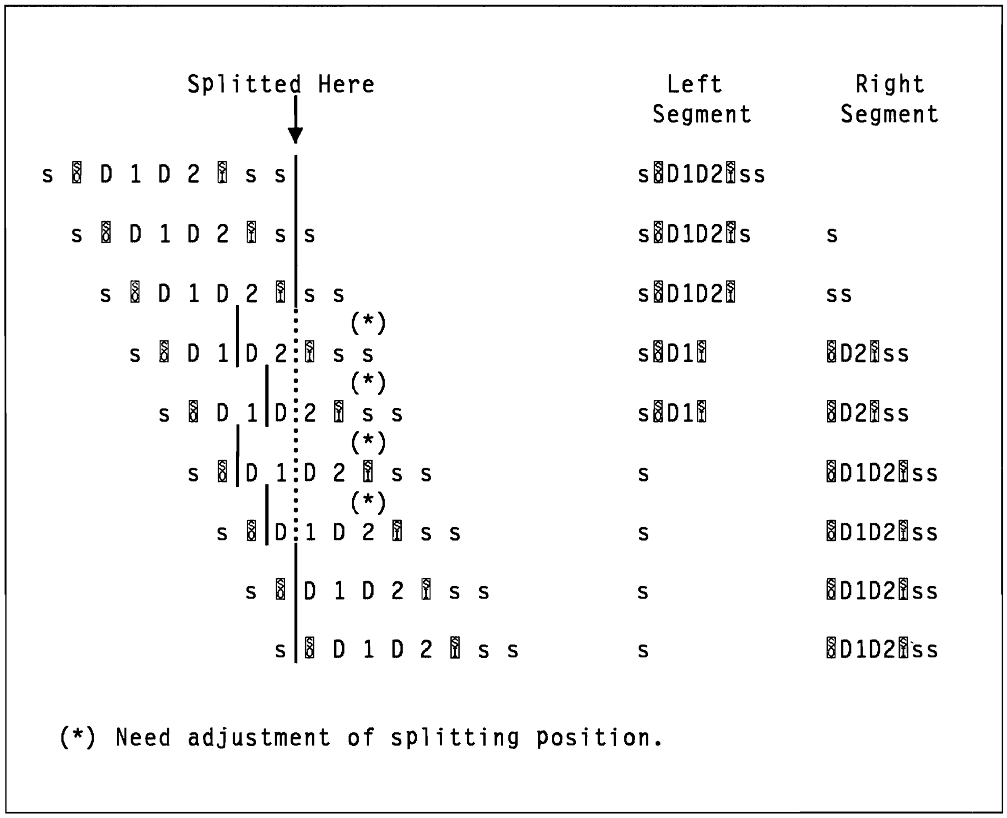
[15] For example, see ISPF scrolling which is described in Appendix B, "Advanced Presentation" on page 89.

```
ss▯D1D2▯ || ▯D3D4▯s → ss▯D1D2D3D4▯s   (SI/SO removal)
                      ss▯D1D2▯▯D3D4▯s (SI/SO no removal)
                      ss▯D1D2ƀD3D4▯s  (SI/SO replaced
                                       with DBCS blank)
```

Figure 11. Example of Concatenation

Splitting

A string should be split at character boundaries so that the substrings results in valid data. When splitting a Mixed string within a DBCS portion, adjustment of the splitting position is required to keep the contents of data in both segments. And the same adjustment as the truncation operation is also required for both segments. It is necessary to add these adjustment logic to the process for breaking or wrapping a string into multiple shorter lines.



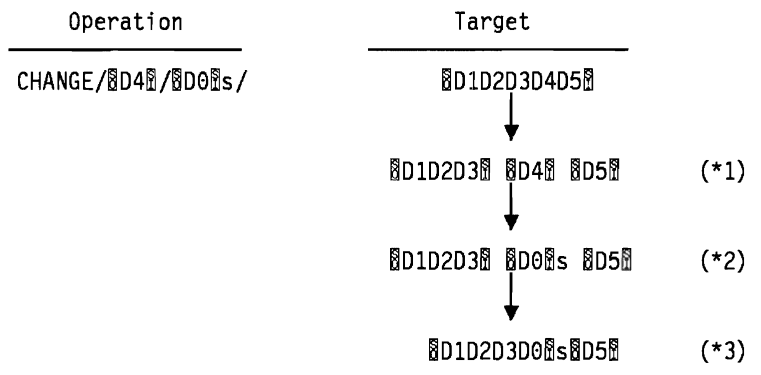
```
          Splitted Here                  Left       Right
                |                       Segment    Segment
                v
s ▯ D 1 D 2 ▯ s s|                      s▯D1D2▯ss
  s ▯ D 1 D 2 ▯ s|s                     s▯D1D2▯s    s
    s ▯ D 1 D 2 ▯|s s                   s▯D1D2▯     ss
               | :   (*)
      s ▯ D 1|D 2:▯ s s                 s▯D1▯       ▯D2▯ss
             |   :   (*)
        s ▯ D 1|D:2 ▯ s s               s▯D1▯       ▯D2▯ss
               | :   (*)
          s ▯|D 1:D 2 ▯ s s             s           ▯D1D2▯ss
             |   :   (*)
            s ▯|D:1 D 2 ▯ s s           s           ▯D1D2▯ss
              s ▯|D 1 D 2 ▯ s s         s           ▯D1D2▯ss
                s|▯ D 1 D 2 ▯ s s       s           ▯D1D2▯ss

(*) Need adjustment of splitting position.
```

Figure 12. Examples of Splitting Mixed String

Change

The change operation requires the combination of some Mixed data processing logic such as search, truncation, and adjacent SO/SI. The typical processing logic of the change operation is shown in Figure 13.



(*1) Search the string to be changed, and extract it from the target string. (need the logic of SPLIT operation)

(*2) Change

(*3) Concatenate each portion removing adjacent SO/SI pairs.

Figure 13. Logic of Change Operation

Validity Check

DBCS strings or Mixed strings must be checked against some criteria before displaying/printing to make sure of the validity. When a validation routine detects an error, some action must be taken. The typical action is either to issue an error return code and stop processing, or to treat the string as a valid one after doing some compensation with/without a warning message. The following shows the typical compensation.[16]

SO, SI pair error replace unpaired SO, SI by supplemental SBCS character.

```
␎D1D2D3    ——→    #D1D2D3      # : Supplemental SBCS character
```

[16] The validity check and compensation routines are provided by some presentation service programs. However, in case that applications should provide them by themselves, follow the rules described here.

Length error drop the last odd byte.

```
␎D1D2D3D␏  ───►  ␎D1D2D3␏
```

Code check replace invalid DBCS codes by supplemental DBCS character.[17]

```
␎D1D2ddD3␏  ───►  ␎D1D2DDD3␏    dd : Invalid DBCS character
                                DD : Supplemental DBCS character
```

Note: As for the validation of DBCS string, follow the first two rules.

3.3 Collating Sequence

A binary value-based sequence is used in SBCS applications. However, it is insufficient and sometimes meaningless for DBCS applications. Then the culture-sensitive ordering is required.[18] For example, Kanji (Japanese) has several sorting algorithms and they are used in the applications.

An application that uses such ordering should have a mechanism for invoking a routine and table to convert a DBCS string into a value ready for binary ordering.

3.4 Mixed/DBCS String Conversion

Conversion of a DBCS string to a Mixed string is an operation which adds SO/SI to the first and last byte positions of the DBCS string. Conversion of a Mixed string to a DBCS string is an operation which deletes the SO/SI if the Mixed string is a DBCS string bracketed with SO/SI. For all other forms of Mixed strings, treat them as an "INVALID CONVERSION" , or convert the SBCS characters to the equivalent DBCS code (42nd ward DBCS)[19] and strip off the SOs and SIs, depending on the user requirements.

Figure 14 on page 23 summarizes the data attribute conversion rule.

[17] The supplemental DBCS character should be selected from the 42nd ward DBCS characters.

[18] The culture-sensitive sequence is defined per DBCS by each country. The sequence for sorting the Mixed strings is not defined yet.

[19] In Japan, there is the conversion of SBCS Katakana to equivalent DBCS Katakana (43rd ward DBCS). In 43rd ward DBCS, the second byte is the same hexadecimal value of a corresponding Katakana code.

```
   D1D2   ──► ␎D1D2␏     converting a DBCS string to
                         a Mixed string

 ␎D1D2␏   ──►  D1D2      converting a Mixed string to
 AB␎D1␏   ──►  .A.BD1    a DBCS string
```

Figure 14. Examples of Mixed/DBCS String Conversion

3.5 Data Type Determination

When developing applications, it is often required to assign a data type to each data item. It is easy to do it in the SBCS environment because only SBCS data is used as character data in SBCS applications. However, under the DBCS environment, since three types of data; DBCS, SBCS/DBCS mixed, and SBCS, are used, it is not so easy to determine a data type for each data item.

For example, in an application for a typical company in Japan, the data type for "employee name" is certainly "DBCS" . Because it can be assumed that all the employees are Japanese, so all the employee names are represented using DBCS characters. However, in the application for an international company in Japan, the data item may be "Character (SBCS)" . Because it can be assumed that Japanese employees coexist with foreigner employees whose names are represented using Latin English.

When developing a DBCS application, developers should always consider which data type is suitable for each data item.

3.5.1 Characteristics of DBCS and Mixed Data Items

If your application is for a specific environment or a specific user, each data type can be easily determined. But if your application is for multiple environments or multiple users, you should take the way that is acceptable for multiple users. So, which is better for users in Asian countries, DBCS or Mixed? The answer is based on the merits and consideration points of each data type.

DBCS data type

- Merits
  - Portability[20]

    Some applications such as cooperative information processing between S/370 and PC environments requires the delivery of user data from S/370 to PC. DBCS data items are effective from the portability point of view, because it is easy to convert a DBCS string under S/370 to that under PC without changing its length. This also enables the logic of DBCS string handling to be compatible between S/370 and PC.

- Consideration points
  - Length of data items

    Since all the DBCSs contain alphanumeric characters (42nd DBCS characters), it is possible that both alphanumeric characters and Kanji, Hanzi, Hanguel characters coexist in a DBCS data item. However, for example, 10 characters (10 DBCS characters) are enough for names of Japanese people, but they are not enough for names of American people. That is, to use a DBCS data type for all data items causes an increase of record length. And it is not preferable for users.
  - Compatibility with SBCS application

    Since a DBCS data item requires a different data type and a data field on a screen from an SBCS application, the compatibility of applications cannot be kept between SBCS and DBCS. Generally the coding (processing logic) can be common between SBCS and DBCS applications excepting the data type declaration. However, applications should take it into consideration that all operations working on a one-byte basis for Character data items will work on a two-byte basis for DBCS data items. This means, the coding of applications containing length-sensitive processing may not be able to be common between SBCS and DBCS.

```
     SBCS Application                       DBCS Application
  ________________________________      ________________________________

  DECLARE VAR1 CHAR(5),                 DECLARE VAR1 GRAPHIC(5),
          VAR2 CHAR(3);                         VAR2 GRAPHIC(3);

  VAR2 = SUBSTR(VAR1,1,3);              VAR2 = SUBSTR(VAR1,1,3);


   VAR1 ← 'ABCDE'  (5 byte)              VAR1 ← 'D1D2D3D4D5'  (10 byte)
   VAR2 ← 'ABC'    (3 byte)              VAR2 ← 'D1D2D3'      (6 byte)
```

[20] "Portability" means the ability to utilize user data and program sources with different operating systems.

Character (Mixed) data type

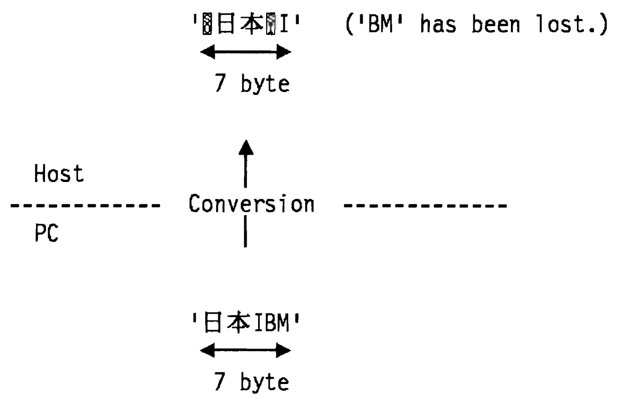
- Merits
  - User interface

    Users can use both SBCS and DBCS characters.
  - Compatibility with SBCS application

    Since both Mixed strings and SBCS strings are stored in Character data items, the coding can be common between SBCS and DBCS as far as it contains just simple move (assignment) operations.
- Consideration points

  The major consideration points for Mixed data items result from the existence of SO/SI characters.
  - Portability of data

    A Mixed string changes its length when ported from a PC environment that does not need shift-codes[21] to a Host environment. So the portability of user data cannot be kept, because some data contents of Mixed data may be lost due to SO/SI insertion at the conversion time.



  - SO/SI positioning on screen

    The appearance of Mixed data may be unacceptable for users when they are displayed in the form of list as shown below. This is because SO/SI characters are displayed as blanks.

```
      Name              Address
  ---------------    -------------
   石井  俊昭           福岡県
   中村  肇             富山県
  Tom Loveland        New York
  Dave Adler          New York
   金子  明史           神奈川県
```

21 Refer to Appendix F, "DBCS-PC" on page 113 for additional information.

- Complicated data handling

  Mixed string handling is more complex than SBCS or DBCS string handling since the applications always have to keep Mixed strings valid during each operation.

As shown above, both DBCS and Mixed data items have some consideration points, while they also have some merits for users. That is, there is no definite answer which is better, DBCS or Mixed.

3.5.2 Customization

In order that applications are used commonly among multiple environments (users), the customization [22] capability is required in some degree. This is not a unique requirement in DBCS countries, but the common one of worldwide users. For example, it is frequently necessary to change data format defined in an application such as data length, etc., according to the corresponding data defined in each user's data base.

Generally, the following items should be customizable to meet the user requirements.

- Data declaration in source code

  Data length of each variable used in an application should be changeable.

- I/O specification

  Layouts of maps should be changeable according to the data length.

- Data definition for data base

  The data length, column name, etc., should be changeable according to data base definition of each user.

In DBCS countries, the customization capability of data types is required in addition to the above items. That is, users require to change data types to either DBCS or Character (Mixed), according to the data base/data file definition. If the data to be handled is the Mixed data, it is not necessary to change the data type (Character data type), however, some additional functions, such as the SO/SI pairing compensation function, are required. To resolve the problem of data types described in the previous section, it is best that applications be customizable, where the proper data type can be defined according to the user requirements.

[22] "Customization" means the ability to redesign a data processing system including program logic, data declaration, etc., to meet the requirements of particular users at the installation or the execution time.

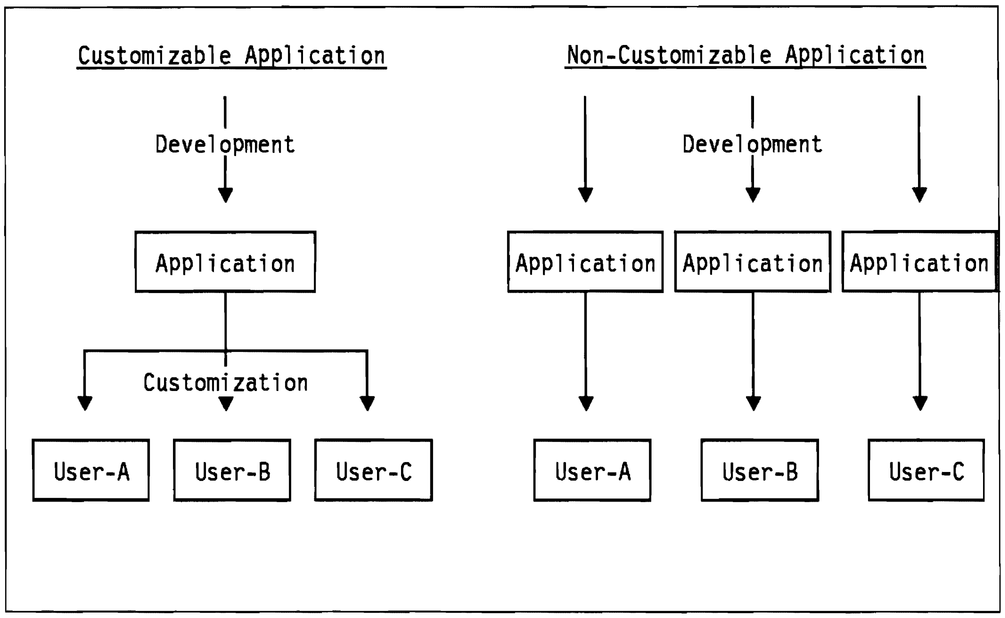


Figure 15. Customizable/Non-Customizable Application

4.0 Presenting DBCS

In presenting DBCS data on I/O devices, most application software makes use of presentation/printer service programs. This chapter provides the total picture of the I/O service in respect with the DBCS data. For details on the programming interfaces and the data streams, refer to 5.0, "DBCS Enabled IBM Key Software" on page 37 or Appendix A, "Device Data Stream" on page 71.

In case of the displays, an application program only specifies the field type in those programming interfaces. If application data is passed to the service program, it then generates an appropriate data stream for the target device.

In case of the system printers and some terminal printers in the batch mode operation, a data set or a spool file is first created by an application program and then processed by a printer service program.

4.1 DBCS Presentation

Display: Each DBCS character occupies two display positions. SO/SI characters occupy one display position respectively. These characters are usually displayed as spaces.

System printer: The presentation of DBCS characters depends on the specified font size. Usually the DBCS font with double of the size of SBCS font is used for simple applications. Whether or not SO/SI characters take print positions depends on the control transferred to print service program (PSF, etc.).

Terminal printer: One DBCS character takes two print positions on terminal printers. Whether or not SO/SI characters take print positions depends on the device type (LU-1 or LU-3)[23] determined by the setting of VTAM.

[23] SO/SI characters take one print position respectively on LU-3 printers, and they do not on LU-1 printers.

SA (Set Attribute) order: In addition to SO/SI control characters, an SA order can be used in the data stream to identify DBCS from SBCS on displays/printers.[24] By using SA order, you can present Mixed string with no space between DBCS and SBCS on a display (LU-2) and a printer (LU-3).[25]

Field outlining (ruled line): Field outlining is mainly used to show data in tabular form or to identify where input fields are located. This function is logically independent of DBCS. But this document includes descriptions on it because the use of outlining is prevalent in DBCS countries, especially in Japan. Field outlining is supported as one of the extended field attributes on displays, and as one of the print functions on printers. Each field outlining is composed of four components; overline, underline, left vertical line, and right vertical line. And these four components can be freely combined (there are 16 combinations in all).

All presentation service products supplied by IBM provides an API that enabled applications to define field outlining[26].

4.2 Display

From the application interface point of view, display I/O is categorized into two modes, "Full screen mode" and "Line mode" .

4.2.1 Full Screen Mode

In full screen mode, data transmission is on the 3270 display field basis. To present DBCS data on display in full screen mode, application programmers must always pay attention in selecting a proper display field according to the type of the data item in the program. The displays of IBM PS/55 provide the following fields;

DBCS field: A DBCS field displays and accepts only DBCS characters. In this field each character is recognized in a double-byte unit.

[24] In the data stream for printers (SCS, etc.), SA is not an order but a control character. The term "SA order" and "SA control character" are sometimes used synonymously for printers in this section.

[25] When preparing data which is presented by the SA order, some special considerations on the data length are required. (Refer to Appendix A, "Device Data Stream" on page 71 for more details.)

[26] Refer to 5.0, "DBCS Enabled IBM Key Software" on page 37.

Mixed Field: A Mixed field displays and accepts both SBCS and DBCS characters. SO/SI characters are required to identify DBCS characters from SBCS characters. SO/SI characters always occupy one screen position, respectively. And they are usually displayed as spaces on displays.

SBCS Field: A SBCS field displays and accepts only SBCS characters. This field is basically equivalent to the field of the current 3270 family.

Figure 16 shows the relationship of inbound (IN) data and outbound (OUT) data to the fields regarding the 3270 data stream.

		FIELD					
		SBCS		DBCS		MIXED	
		IN	OUT	IN	OUT	IN	OUT
DATA	SBCS	Y	Y	N	N	Y	Y
	DBCS	N	N	Y	Y	N	N
	MIXED	N	Y*	N	N	Y	Y

Y = Yes N = No

Figure 16. Relationship of Inbound Data and Outbound Data to Field

Note: (*) The SBCS field accepts a Mixed string in an outbound data stream.

The following presentation service products provide programming interfaces for the fields described above.[27]

- ISPF/DM[28]
- GDDM
- CICS/BMS
- IMS/MFS

[27] Refer to 5.0, "DBCS Enabled IBM Key Software" on page 37 for some considerations for the API provided by these service programs.

An Application program can present data in full screen mode without these presentation services. The CONSOLE macro or the DIAG code X'58' instruction with CCW code X'29' (CMS environment) and the TPUT/TGET macros with FULLSCR option (TSO environment) are used for this purpose. In this case, the application program must generates the 3270 data stream by itself.

[28] The VSE version of ISPF does not provide the interface to DBCS fields.

4.2.2 Line Mode

In line mode, data transmission is based on the sequential transmission of lines, which is usually from the leftmost column to the rightmost column on the display. The following products provide a programming interface for line mode I/O.

- CICS
- TSO
- CMS
- ICCF

The chart below shows overview of the application programming environment regarding the presentation on a display.

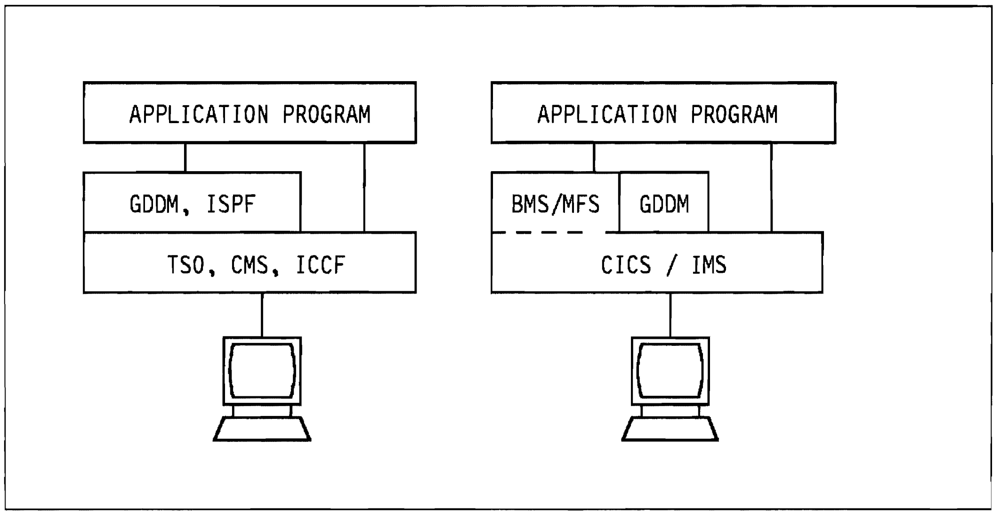


Figure 17. Software Environments for Display

Note: Figure 17 shows the typical case of the presentation on displays.

4.3 System Printer

Line print support, similar to the line mode presentation on a display, is provided for all DBCS printers. For AFP (Advanced Function Printing) printers, page print functions using the APA (all-points-addressable) function provided by print service programs are DBCS enabled.

4.3.1 Line Print

To print DBCS data on system printers such as IBM 3800, 3820, 3827, or 3835 printer in line mode, there is no special consideration other than the validation of

DBCS data to be printed. The printer service programs such as PSF (Print Services Facility) are enabled to handle DBCS data properly.[29]

Print format: When printing data in line print mode, application programs generate a data set or a spool file with a proper print format. There is no specific print format for DBCS data as long as DBCS characters are enclosed with SO/SI control characters. SO/SI control characters must be paired within each logical record.

You can also include information specifying field outlining (ruled line) in a data set or a spool file. The field outlining function is implemented by using SA (Set Attribute) control characters. The format of the SA control character is similar to that used in a SNA Character String (SCS). Refer to Appendix A, "Device Data Stream" on page 71 for more details.

Note: Although the SA control character specifying field outlining is not supported by PSF, Kanji Print Utility version 2.1 or later can convert it to a special syntax for PSF.

The following chart shows the overview of DBCS line print functions.

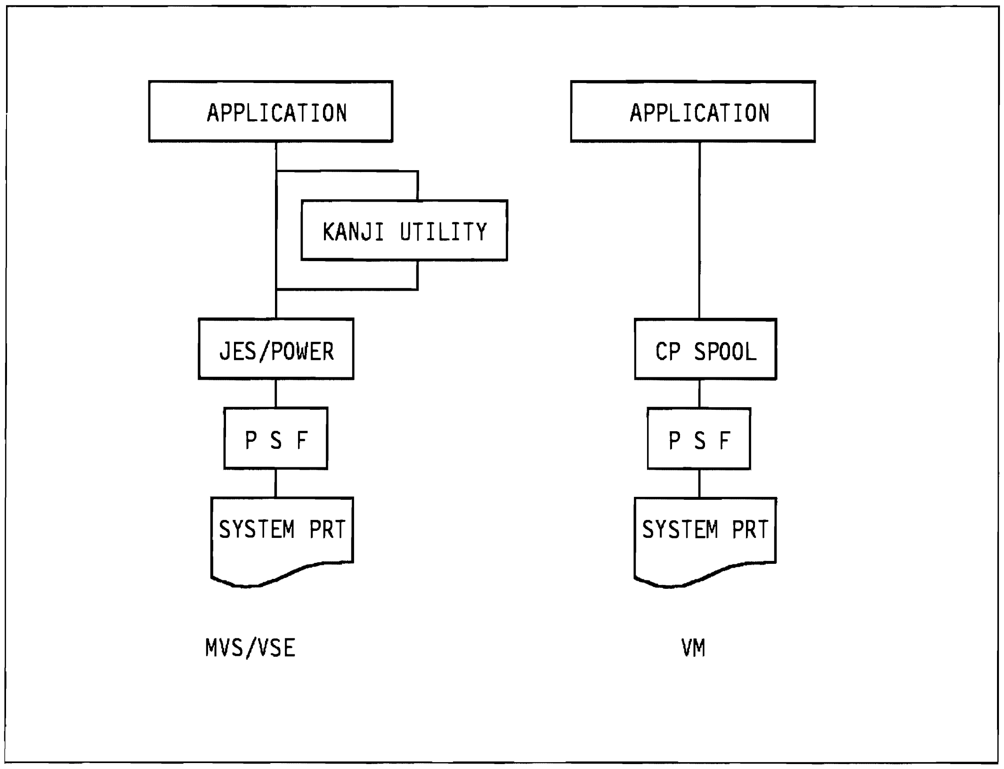


Figure 18. Overview of DBCS Line Print Functions

Note: Under the VM environment, the Kanji Print Utility is not available. This prevents the use of field outlining.

[29] PSF is a prerequisite software when printing DBCS data on system printers.

4.3.2 Page Print

A formatted data set or a spool file can be printed on a system printer by the page definition functions of the AFP software. Information on the fields and fonts, which are specified in separate steps, is referred to by PSF. The page definition function is used to make an advanced print-out which requires, for example, mapping of fields, and various kinds of fonts.

The page definition function is provided by PMF, PPFA, and PSAF.

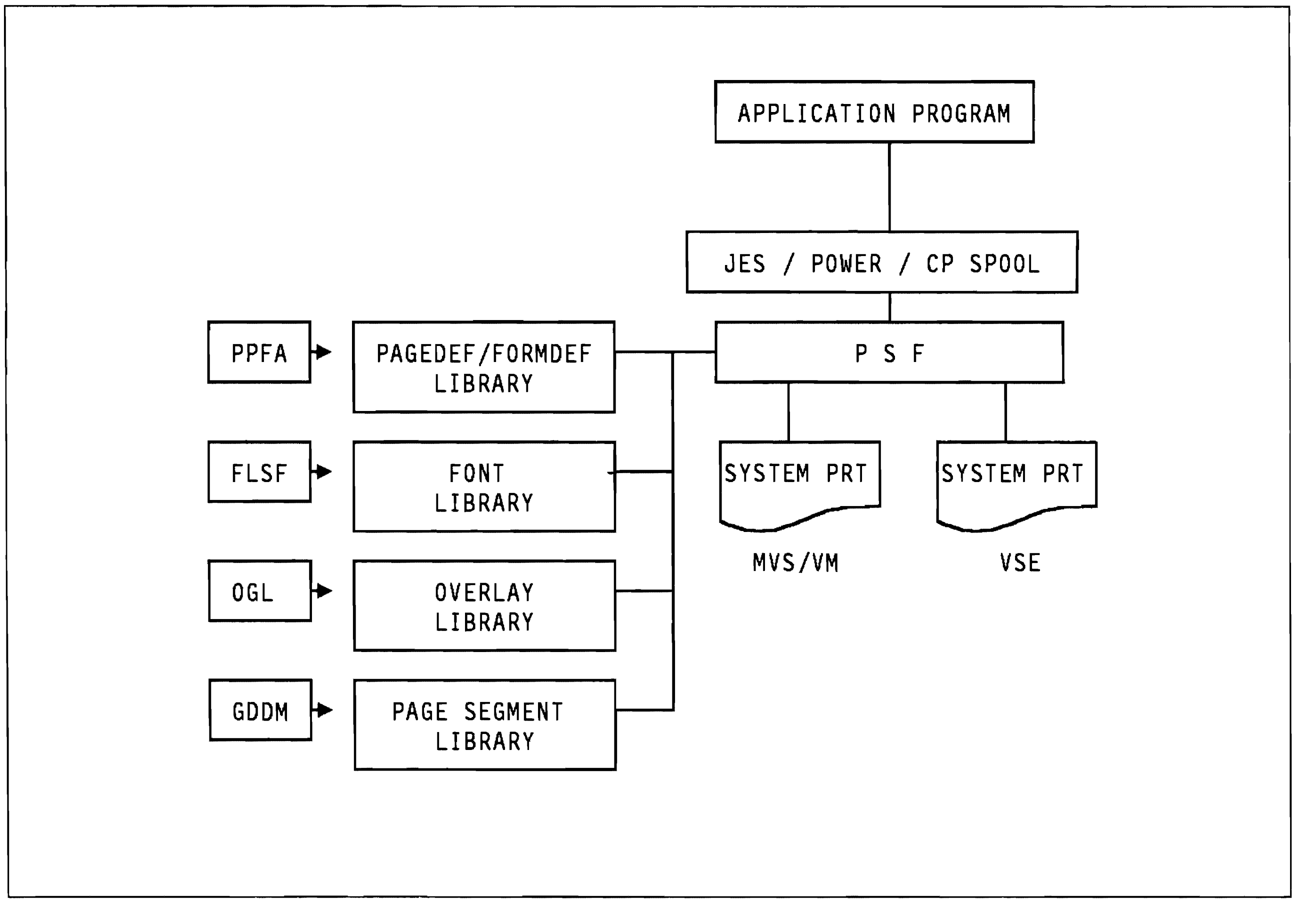


Figure 19. Overview of Advanced Function Printing

4.4 Terminal Printer

There are two types of services for DBCS printing on IBM PS/55 attached printers. One is the programming interface of the presentation services provided by IMS or CICS. It is similar to the programming interface for the display. Another is the service provided by the printer driver program such as KDP (Kanji Data Set Print Program) or RSCS (Remote Spooling Communications Subsystem). The terminal printer supports only the line format data, as described for the system printers.

The figure below shows the overview of terminal printers.

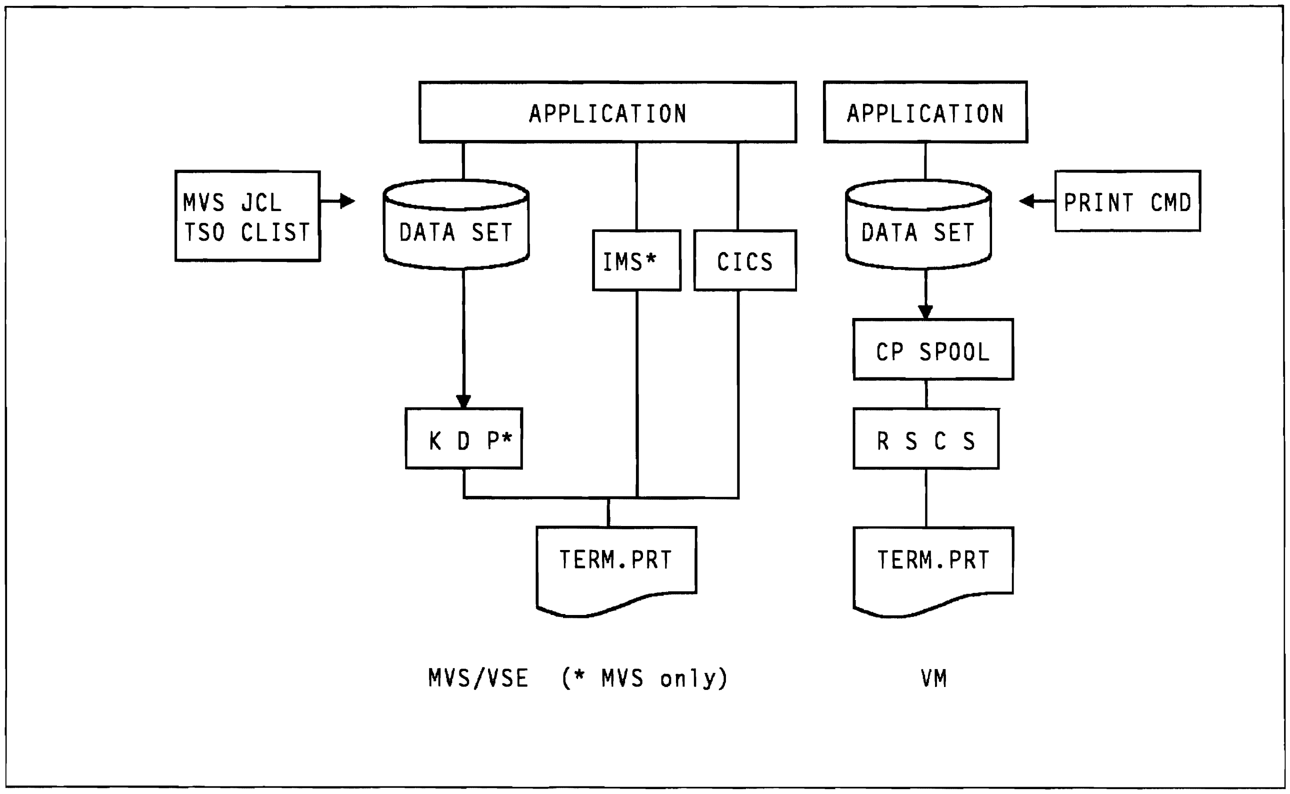


Figure 20. Software Environment for Terminal Printers

Note: Both KDP and RSCS can handle SA control characters imbedded in the print data to draw field outlining (ruled line).

5.0 DBCS Enabled IBM Key Software

IBM offers various programs/utilities that will improve the productivity when writing DBCS application programs. By using these programs, application programs can manipulate/present DBCS data easily. In this chapter, actual programmings interfaces of the following products and/or considerations are described in terms of DBCS support.

- Presentation services[30]
  - ISPF/DM
  - GDDM
  - CICS/BMS
  - IMS/MFS
- Application generator/Map Definition
  - CSP
  - SDF II
- Printer services
  - PSF
  - Kanji Utility
  - Kanji Data Set Print Program
  - RSCS
- Data base managers
  - DB2
  - SQL/DS
  - IMS/DB, DL/I
- Languages
  - COBOL
  - PL/I
  - FORTRAN
  - Assembler

[30] DBCS applications are required not only to present DBCS data but also to present data with various extended attributes such as field outlining. In this section, how to define fields with extended attributes is described as well as how to define DBCS and Mixed fields on a display.

5.1 ISPF/DM

ISPF (Interactive System Productivity Facility) is one of the presentation services products, and provides functions that simplify interactive application development. ISPF version 2 or later supports DBCS.

When handling DBCS data as texts on panels, messages, or user data, the following are to be considered.

5.1.1 DBCS Panel Definition

In order to handle DBCS data on panels, proper fields types should be specified according to the type of DBCS data. The field types can be defined in the attribute section of each panel definition.

The following keywords in the attribute section related to DBCS are provided:

- FORMAT(EBCDIC | DBCS | MIX)
- COLOR(value)
- HILITE(value)
- OUTLINE([L] [R] [O] [U] | BOX | NONE)

Note: You can define extended field attributes by specifying COLOR, HILITE, or OUTLINE keywords.

1. Input field

   Using the FORMAT(DBCS | MIX) keyword, DBCS and Mixed fields can be defined. DBCS fields must have an even length.

   ```
   < Example >

     )ATTR
     @   TYPE(INPUT) FORMAT(MIX)    /* define Mixed input field */
     *   TYPE(INPUT) FORMAT(DBCS)   /* define DBCS input field  */
   ```

2. Output field / Text

   Similarly to input field, either DBCS field or Mixed field can be defined with FORMAT(DBCS | MIX) keyword.

   Note: Mixed strings can be displayed in a field specified with no FORMAT parameter, however, they can not be displayed in a field where FORMAT(EBCDIC) is specified.

5.1.2 Data Validation

ISPF provides the functions which validate DBCS data under some criteria when DBCS data are sent to terminals.

SO/SI pairing: ISPF converts unpaired SO/SI characters to periods (.). The string is presented as an SBCS string.

Length: For a Mixed string, the length of the DBCS portion is checked. If it is odd, SO/SI characters are converted to periods (.), so the string is presented as an SBCS string. For a DBCS string, the substitute DBCS character (X'4195') replaces the last odd byte making the length one byte longer than the original length.[31]

Code range: The substitute DBCS characters replace DBCS characters which have invalid code points.

5.1.3 Built-in Functions for DBCS data

ISPF provides the following built-in functions for DBCS processing:

ADDSOSI	Adds SO/SI to a DBCS string, (converting a DBCS string to a Mixed string).
DELSOSI	Deletes SO/SI from a bracketed DBCS string, (converting a bracketed DBCS string to a DBCS string).
ONEBYTE	Converts DBCS characters to the corresponding SBCS characters.
TWOBYTE	Converts SBCS characters to the corresponding DBCS characters.

The TWOBYTE function can be used as the argument of the ADDSOSI function, and the DELSOSI function can be used as the argument of the ONEBYTE function. The combination of these functions are used for the conversion between a SBCS string and a Mixed string (bracketed DBCS string).

```
variable = ADDSOSI(TWOBYTE(variable name))
                    ... Converts SBCS to Mixed string

variable = ONEBYTE(DELSOSI(variable name))
                    ... Converts Mixed (bracketed DBCS) to SBCS string
```

Note: Whether the lower-case alphabetics are converted by the TWOBYTE function to DBCS alphabetics or to DBCS Katakana, depends on the translation table of ISPF[32].

Only DBCS alphanumerics (42nd ward DBCS characters) are converted by the ONEBYTE function.

[31] As far as a DBCS string is stored as a DBCS data item, the length error of the DBCS string does not occur.

[32] When "3278KN" is specified as the terminal type, the translation table for Katakana characters is used.

5.1.4 Installation of DBCS Panel/Message

ISPF has two kinds of libraries for panels and messages. ISPPLIB/ISPPALT for panels and ISPMLIB/ISPMALT for messages. The DBCS panels and messages should be allocated in ISPPALT and ISPMALT, respectively.

Note: ISPPLIB and ISPMLIB are the primary libraries for panels and messages. ISPPALT and ISPMALT are the alternate ones for DBCS users. When the terminal has DBCS capability, panels/messages in ISPPALT/ISPMALT are sent to the terminal. Otherwise, those in ISPPLIB/ISPMLIB are sent. ISPF has the capability to query the terminal characteristics.

The following are examples which are used to concatenate an application's and ISPF's panels/messages into ISPPALT and ISPMALT.

```
< Example >

      1. JCL (MVS)
            :
         //ISPPALT   DD DSN=xxxx.xxx,VOL=SER=.....,
         //             DISP=SHR
         //          DD DSN=SYS1.ISPPALT,VOL=SER=....,
         //             DISP=SHR
         //ISPMALT   DD DSN=yyyy.yyy,VOL=SER=.....,
         //             DISP=SHR
         //          DD DSN=SYS1.ISPMALT,VOL=SER=....,
         //             DISP=SHR
            :

      2. CLIST (MVS)
            :
        ALLOC F(ISPPALT) SHR REU DSN('xxxx.xxx','SYS1.ISPPALT')
        ALLOC F(ISPMALT) SHR REU DSN('yyyy.yyy','SYS1.ISPMALT')
            :

      Note: 'xxxx.xxx' is the name of data set including
            an application's panels. 'yyyy.yyy' including
            messages.


      3. REXX (VM)
            :
        'FILEDEF ISPPALT DISK xxxxx xxxxx x      (PERM'
        'FILEDEF ISPPALT DISK ISPPALT MACLIB fm (PERM CONCAT'
        'FILEDEF ISPPMLT DISK yyyyy yyyyy y      (PERM'
        'FILEDEF ISPPMLT DISK ISPMALT MACLIB fm (PERM CONCAT'
            :
```

5.2 GDDM

GDDM (Graphic Data Display Manager) is a family of IBM products that make it possible for application programs to present graphics, alphanumerics, and images on displays, printers and plotters, and to read input from displays. It provides the DBCS functions for generating full screen panel definitions with the DBCS data handling capability. Note that, specification of "MIXSOSI=YES" in the defaults file (ADMADFx) is required to enable DBCS data handling in GDDM.

5.2.1 DBCS Panel Definition

The following calls are used to define fields which have the extended field attributes.

ASDFLD Sets a field. Basic field attributes except for intensity can be defined.

ASFPSS Sets a character set. The DBCS field can be defined with this call.
ASFPSS(field-id,248) ... DBCS field

Note: DBCS fields must be defined with an even length.

ASFSEN Mixed field can be defined with this call.
ASFSEN(field-id,-1) ... Current setting
ASFSEN(field-id, 0) ... EBCDIC field
ASFSEN(field-id, 1) ... Mixed field with SO/SI
ASFSEN(field-id, 2) ... Mixed field with SA order (GDDM V2.1 up)

Note: An EBCDIC field accepts Mixed data for display, but cannot be used for Mixed data input.

ASFBDY Sets field outlining.
ASFBDY(field-id,-1) ... Current setting
ASFBDY(field-id, 0) ... No field outlining
ASFBDY(field-id, n) ... Set field outlining (n : 0-15)[33]

ASFCOL Sets extended colors.

ASFHLT Sets extended highlighting.

ASFTRA Sets transparent characteristics.

The ASRFMT call is used to specify the multiple field attributes described above in a single invocation.

[33] 16 different combinations of field outlining can be set by specifying n. Specify ASFBDY(field-id,15) for a field with box field outlining. Refer to Appendix A, "Device Data Stream" on page 71 for more information.

5.2.2 Query Terminal Characteristics

When DBCS data is to be displayed, an application program should know whether the terminal has DBCS capability. The "FSQURY" call is used to get the information on the terminal characteristics.

```
< Format >

    FSQURY(0,1,18,ARRAY) (ARRAY is an array of full word integers.)

< Returned information related to extended attributes >

   ARRAY( 1) ... Device family ('1' indicates display family.)
   ARRAY(12) ... Number of colors supported
   ARRAY(13) ... Highlighting capability
   ARRAY(16) ... Background transparency support
   ARRAY(17) ... DBCS support
                  '0' - No DBCS support
                  '1' - DBCS support
   ARRAY(18) ... Field outlining support

< Example to use FSQURY ( PL/I ) >
        ...........
    DECLARE  ARRAY(18)  FIXED BINARY(31);
    CALL FSQURY(0,1,18,ARRAY);
        ...........
```

5.2.3 Data Validation

GDDM provides the functions which are used to validate DBCS data under some criteria.[34]

SO/SI pairing: GDDM converts unpaired SO/SI characters to blanks. The string is presented as an SBCS string.

Length: For a Mixed string, the length of the DBCS portion is checked. If it is odd, blanks replace SO/SI characters, and the string is presented as an SBCS string.

Code range: The DBCS blanks replace the DBCS characters which have invalid code points.

In addition to the functions above, GDDM converts SO/SI characters to blanks when Mixed strings are sent to a non-DBCS terminal.

[34] Refer to "Validity Check" on page 21 for the detailed criteria.

5.3 CICS/BMS

BMS (Basic Mapping Support) of CICS (Customer Information Control System) provides the mapping support for the display/terminal printer under the MVS and VSE environments. The CICS/OS/VS Version 1 Release 7.0 - BMS provides the DBCS data handling capability, which includes the DBCS field and the MIXED field support. Field outlining is also supported.

5.3.1 DBCS Map Definition

DFHMSD macro

Defines a map set. The generated map sets are stored in the CICS load library under the user-defined names. By using DFHMSD macro, the properties of maps can be controlled by map set unit.

```
DFHMSD      :
         ,MAPATTS=(attr1,attr2, ...)
         ,DSATTS=(attr1,attr2, ...)
        (,EXTATT=[ NO | MAPONLY | YES ])
            :
```

MAPATTS and DSATTS operands specify the attributes types to be included in physical maps or DSECTs (symbolic description maps)[35] respectively. These types can be COLOR, HILIGHT, OUTLINE, PS, SOSI, TRANSP, and VALIDN. This list must include all the attribute types to be specified for individual fields in the map.

Note: Any type included in DSATTS should also be included in MAPATTS.

Note: EXTATT operand is supported for compatibility with previous release. For new releases, the MAPATTS and DSATTS operands should be used instead.

"EXTATT=NO" disables both MAPATTS and DSATTS operands.

"EXTATT=YES" is equivalent to:

```
MAPATTS=(COLOR,HILIGHT,PS,VALIDN)
DSATTS=(COLOR,HILIGHT,PS,VALIDN)
```

"EXTATT=MAPONLY" is equivalent to:

```
MAPATTS=(COLOR,HILIGHT,PS,VALIDN)
```

[35] The attribute types included in DSECTs can be modified dynamically by applications at the execution time.

DFHMDI macro

Defines a map. You can set the default attributes of the fields within the map.

```
DFHMDI      :
        ,MAPATTS=(attr1,attr2, ...)
        ,DSATTS=(attr1,attr2, ...)
       (,EXTATT=[ NO | MAPONLY | YES ])
          :
```

The use of these operands is the same as the DFHMDS macro. Refer to the description above.

DFHMDF macro

Defines fields within a map. The following operands are used to specify extended attributes;

```
DFHMDF      :
        ,COLOR=[ DEFAULT | color ]
        ,EXTATT=[ NO | MAPONLY | YES ]
        ,HILIGHT=[ OFF | BLINK | REVERSE | UNDERLINE ]
        ,OUTLINE=[ (LEFT,RIGHT,UNDER,OVER) | BOX ]
        ,PS=[ BASE | psid ]
        ,SOSI=[ NO | YES ]
          :
```

Specify "PS=F8" for a DBCS field and "SOSI=YES" for a MIXED field.

5.3.2 DBCS/SBCS Map Switching

Some application programs are required to support both SBCS and DBCS terminals within a single transaction (refer to 6.3, "Multilingual Support" on page 68). This requirement can be easily satisfied using the BMS map set suffixing. That is, the BMS map set suffixing can be applied to the automatic switching between SBCS and DBCS maps according to the terminal in use.

The suffixed names of maps can be specified using the SUFFIX operand of DFHMSD macro. The suffix is of your own choice, however, the same suffix must also be specified in the Terminal Control Table (TCT) using the ALTSFX operand in advance.

5.3.3 Query Terminal Characteristics

The CICS provides the capability for terminal querying. Applications can get the characteristics of terminals using the ASSIGN command. The following are the options related to DBCS and extended data stream.

SOSI	shows the SO/SI creation capability.
OUTLINE	shows the field outlining capability.
COLOR	shows the extended color capability.
EXTDS	shows the extended data stream capability.
HILIGHT	shows the extended highlighting capability.
PS	shows the programmed symbols capability. (shows DBCS filed capability.)
KATAKANA	shows the terminal is a Katakana terminal.[36]

Note: The returned value X'FF' shows that the terminal has the corresponding capability. And the value X'00' shows no capability.

The terminal which has these capabilities should be correspondingly defined in the Terminal Control Table (TCT).

5.3.4 Data Validation

CICS does not validate the data sent to terminals. An application program should check the validity of the DBCS data before sending it to terminals.

5.3.5 Literal/Constant

DBCS or Mixed literals cannot be used as the initial values of the field. Use the hexadecimal format literal ("XINIT" operand) or SDF II (Screen Definition Facility II) for panel design.

[36] This option (KATAKANA) is used for Japan only. Refer to 6.4, "Katakana Terminal Support" on page 69 for more information.

5.4 IMS/MFS

MFS (Message Format Service) of IMS (Information Management System) provides the mapping support on terminals for MVS environments. Version 1 Release 3.0 and Version 2 Release 1.0 provide DBCS support as the "5550 Support Feature" (called "Kanji Enhancement" in Japan), which is a no-charge feature available for Asian countries. This feature is incorporated into the base code of the IMS/VS Version 2 Release 2.0 or later.[37] The IBM PS/55 display and printer should be defined by TYPE and TERMINAL macros in the IMS/VS System Definition as follows:

```
< IBM PS/55 Display >

      TYPE      UNITYPE=SLUTYPE2,
                   :
      TERMINAL  TYPE=3270,
                MODEL=2,       or   TYPE=3270-A02
                   :

< IBM PS/55 Printer >

      TYPE      UNITYPE=SLUTYPE1,
                   :
      TERMINAL  COMPT1=MFS-SCS1,
                   :
```

5.4.1 DBCS Map Definition

Mixed and DBCS fields are defined by the "EATTR" parameter of the "DFLD" (Define Field) statement. This statement is also used to specify a Mixed string to be printed on terminal printers.

The IMS/MFS also provides the field outlining (ruled line) capability by generating the SFE order for displays and the SA control character for terminal printers. Field outlining is also defined by the "EATTR" parameter.

[37] This feature generates a 3270 Extended Data Stream for the IBM PS/55 display, and a SNA Character String for an IBM PS/55 attached printer.

You can specify the following values to designate the form of the field outlining.

```
       ┌ OUTL       (used for dynamic attribute modification)
       │ OUTL'nn'   (nn = X'00' - '0F')
       │ BOX
EATTR= │ RIGHT
       │ LEFT
       │ UNDER
       └ OVER
```

```
[label]    DFLD  EATTR=(... ,OUTL'0F')

[label]    DFLD  EATTR=(... ,RIGHT,LEFT,OVER)
```

Display

- Mixed field

  ```
  [label]    DFLD  EATTR=(... ,MIX)

  [label]    DFLD  EATTR=(... ,MIXD)
  ```

  Note: EATTR = MIXD is used for the dynamic attribute modification.

- DBCS field

  A DBCS field is defined by specifying EGCS in the "EATTR" parameter list.

  ```
  [label]    DFLD  EATTR=(... ,EGCS)   (or EGCS'nn')

             'nn' specifies the LCID to be set.
             When 'nn'='F8', DBCS field is defined.
  ```

  The initial values of DBCS fields can be specified using DBCS literals. The format of DBCS literal is:

  ```
        G'▨D B C S▨'
  ```

Terminal Printer (IBM PS/55 attached printer)

```
[label]    DFLD  EATTR=(..., MIX)       (or MIX'nn')

[label]    DFLD  EATTR=(..., MIXS)      (or MIXS'nn')
```

EATTR = MIX is used to print Mixed strings with SO/SIs as spaces (SO/SI print option). EATTR = MIXS is used to print Mixed strings without SO/SIs (SO/SI suppress option).

Note: "nn" is the expected number of SO/SI pairs.

5.4.2 Data Validation

The IMS/MFS validates Mixed strings as follows:

SO/SI pairing: The unpaired SO and duplicate SI characters in Mixed strings to be sent to display/printer are checked. In case that the SO/SI print option is specified, unpaired SO and duplicate SI characters are replaced with blanks (X'40') with the exception of the last unpaired SO character within the field. A SI character is padded at the end of the field for the unpaired SO which is found lastly within the field. When the unpaired SO is at the end of a field, a blank replaces it. In case that the SO/SI suppress option is specified, unpaired SO and duplicate SI characters are removed.

Length: If the length of a DBCS portion is odd, a blank (X'40') is padded after the last odd byte.

Code range: The IMS/MFS does not validate the code range of characters.

Note: Mixed strings are checked only when they are sent to fields defined by EATTR=MIX, MIXD or MIXS. When Mixed strings are sent to SBCS (EBCDIC) fields, they can be displayed/printed correctly as long as they are a valid string, however, no validity check is done.

5.5 CSP/AD, CSP/AE

CSP/AD (Cross System Product/Application Development) is an application generator that provides an interactive interface for defining, testing, maintaining, documenting, and generating application programs to execute in various system environments. And CSP/AE (Cross System Product/Application Execution) provides the capability to execute any application defined and generated by CSP/AD. CSP provides the various functions for DBCS handling and presentation.

5.5.1 Record definition

CSP provides DBCS and MIX data type so that DBCS and Mixed strings can be stored and handled correctly. MIX data type is a data type that has added functions for Mixed strings such as validity check and recovery function to CHAR data type.

```
                 :
     NAME      LEVEL OCCURS TYPE  LENGTH DEC   BYTES  DESCRIPTION
***                     TOP OF LIST
001 RECORD1     10   00001  CHA     00010       00010  ..............
002 RECORD2     10   00001  DBCS    00010       00020  ..............
003 RECORD3     10   00001  MIX     00010       00010  ..............
                 :
```

5.5.2 Map Definition

In order to define DBCS maps, the specification of the proper device type is required during the device selection time. Specify "5550D" for DBCS workstations and "5550P" for DBCS printers.

As for CODES (attribute characters) related to DBCS/Mixed fields, the following characters are available when the above device types are specified;

KCC(+)	DBCS constant field
KCV(@)	DBCS input field
MCC(%)	Mixed constant field
MCV(¦)	Mixed input field

Note: "INIT" command is required to set initial values to DBCS/Mixed fields.

5.6 SDF II

SDF II (Screen Definition Facility II) is a tool that provides the capability for designing, testing, and maintaining panels interactively. SDF II generates ISPF panel definitions for ISPF applications, GDDM/IMD panel objects for GDDM/IMD applications, CICS/BMS macros and associated DSECTs (symbolic description maps) for CICS applications, IMS/MFS utility statements for IMS applications, and CSP/AD panels for CSP applications. These various map objects are generated with the single common user interface. You can specify all attributes supported by the presentation service programs.

SDF II simplifies the specifying of DBCS strings as initial values of literals/constants in panels. Especially, it is useful for CICS applications that DBCS and Mixed strings can be directly specified as the initial value of the fields in BMS panels with SDF II. These strings are converted to XINIT operands (hexadecimal format literals) and placed in the CICS/BMS macros.

SDF II also provides other various functions that enable users to test, and maintain panels interactively. The productivity for developing panels will be improved with SDF II.

5.6.1 Map Definition

In order to define DBCS maps, the specification of the proper device type is required during the device selection time. Specify "5550" as the device type on the "IDENTIFY PANEL" for DBCS maps.

As for MARKS (attribute characters) related to DBCS, the following characters can be used on "DEFINE MARKS" panel;

MI	Define Mixed field
DB	Define DBCS field
BOX	Set field outlining (BR/BL/BO/BUN)

Note: The "INIT" command is required to set initial values to DBCS/Mixed fields.

5.7 PSF

Print Services Facility (PSF) is the print service program which supports Advanced Function Printing (AFP) printers. This program is used as the output writer or as the access method to process output data sets for the following printers.

- IBM 3800 Printing Subsystem Model 8
- IBM 3820 Page Printer
- IBM 3827 Page Printer
- IBM 3835 Page Printer

PSF is a prerequisite for printing DBCS data on the above printers. Before PSF receives a file containing DBCS data, the following should be assured by the application program or user:

- DBCS data are enclosed by SO/SI
- SO/SI pairing is completed in each logical record

DBCS Consideration

1. PRMODE parameter

   When the data set to be printed includes DBCS data, "PRMODE=SOSI1" or "PRMODE=SOSI2" must be specified. The PRMODE parameter determines whether or not SO/SI characters occupy the printing position. SO/SI occupy printing positions when PROMODE=SOSI1 is specified, and no position when PRMODE=SOSI2 is specified. This parameter can be specified:

   - in OUTPUT statement under MVS environment.
   - as command option under VM environment.
   - as control parameter in the Printer-Parameter Member under VSE environment.

2. Field outlining (ruled line)

   If printing data include SA control characters which represent field outlining (ruled line), these control characters must be converted to a special format which PSF can recognize. This conversion can be performed by the Kanji Utility, which is described in the next section.

5.8 Kanji Utility (Kanji Print Utility)

The Kanji Print Utility[38] is a utility included in Kanji Utility Products, and runs under the MVS and VSE environments. This utility should be used when files printed to a system printer contains fixed formatted records with DBCS fields or specifications for field outlining (ruled line).

This utility converts the following data sets into the form which can be printed through PSF.

- KANJI DATA FILE

  Each record in this data set has one or more fields which are previously defined. It is possible to edit each field before printing. DBCS portions defined as DBCS fields need no SO and SI control codes.

- SOURCE STATEMENT DATA SET

  This is the data set containing the source statements of COBOL, PL/I, or etc. Each record in this data set should be of fixed length (80 bytes). DBCS data need to be enclosed with SO and SI. This type of data set can be printed out directly through PSF without this utility.

- SYSOUT DATA SET

  This data set has the print control character (ANSI control character) at the beginning of each record. DBCS data should be enclosed with SO and SI. This type of data set can be printed without this utility unless it includes any specification of field outlining.

- USER DATA SET

  KDP processes this data set in almost the same way as SYSOUT data set except for regarding the first byte of each record as a printable character.

Field outlining is specified with the SA control character. The utility converts the control character to the data format which PSF can process. The SA control character has the same format as that of SCS (SNA Control String). This converting function is supported for SYSOUT and USER data sets.

[38] The program numbers of Kanji Utility are 5799-BWM (MVS version) and 5799-CAQ (VSE version).

The following are statements which are used to specify the information about the data set.

Control Statement	Description
INCTL	Specifies the type and classification of the input data set.
FNTCTL	Specifies the character font information
PRTCTL	Specifies the print control information.
TTLCTL	Specifies the title data of each page.
RECCTL	Specifies the selection control of records.
FLDCTL	Specifies the edit information of records.

The following is a sample JCL which is used to print a USER data set.

```
  //         JOB  ..........
( //JOBLIB   DD   ......... )
( //JOBCAT   DD   ......... )
  //         EXEC PGM=KNJPRUTL                                  (*1)
( //STEPLIB  DD  .......... )
( //STEPCAT  DD  .......... )
  //SYSPRINT DD  SYSOUT=*
  //SYSUT1   DD  DSN=xxxxx.xxxxx,DISP=SHR                       (*2)
  //OUTPUT1  OUTPUT CHARS=(aaa,knj,bbb),DEST=xxxxxxxx           (*3)
  //SYSUT2   DD  SYSOUT=*,OUTPUT=*.OUTPUT1
  //SYSIN    DD  *
    INCTL  PRINT(USER);                                         (*4)
    FNTCTL FONT(aaa) STYLE(1) TYPE(EB);
    FNTCTL FONT(knj) STYLE(2) TYPE(GC);
    FNTCTL FONT(bbb) STYLE(3) TYPE(EB);
    PRTCTL EBFONT(aaa) GCFONT(knj) NOHEAD NOSOSI;
  /*
  //
```

(*1) The program name of the Kanji Print Utility is "KNJPRUTL" .

(*2) "xxxxx.xxxxx" is the name of the data set to be printed.

(*3) In CHARS option, "aaa" and "bbb" are the SBCS font names. "knj" is the DBCS font name.

(*4) The control statements are put in this section.

5.9 Kanji Data Set Print Program

The Kanji Data Set Print Program (KDP)[39] provides the function for printing DBCS characters and drawing field outlining (ruled line) on terminal printers under the MVS/SP environment. Field outlining is represented by the SA control character, whose format is the same as that of SCS (SNA Character String).

Note: This program supports LU-1 printers but does not support LU-3 printers.

There are various options available to control the printing. These options can be specified in the "SYSIN" data definition. The following shows the DBCS related and other key word parameters.

Function Statement	Parameter	Description
INFILE	DSNAME MEMBER	The data set to be printed out is specified in the INFILE statement.
PRINT	COPY FORMAT LUNAME EDIT OUTLINE	Specifies the number of copies. Specifies the print foramt name.(*) Specifies the logical printer name. Specifies the format information of records.(**) If OUTLINE=IGNORE is specified, the SA control characters are treated as part of the user data.
RELEASE	LUNAME BKPAGE	The RELEASE statement can be used to resume printing the suspended data set.
CANCEL	REQNUM	The CANCEL statement can be used to cancel the job.

(*) The print format names are determined at the installation time of the Kanji Data Set Print Program. Each print format includes various format information such as page length, margins, and character size.

(**) The EDIT parameter can be used to specify the attributes of each portion within a record as follows;

```
EDIT(type:pos:len .....)        (pos : start position, len : length)
```

DBCS data without SO/SI can be printed by specifying this parameter. Specify D or K type for DBCS portions, and M or E for Mixed portions.

[39] The program number is 5799-BTG.

The program is invoked by using MVS JCL. The following is a sample JCL which is used to invoke the Kanji Data Set Print Program and the sample "SYSIN" data.

```
< MVS JCL >

 //jobname   JOB ...
 //setp1     EXEC PGM=JPRNFPR,PARM='...'
 //STEPLIB   DD DSN=load-module-library
 //STEPCAT   DD DSN=catalog
 //SYSPRINT  DD SYSOUT=X
 //JPRQUE    DD DSN=request-queue-file
 //SYSIN     DD *
 ******************************************************
 * THIS IS A SYSIN DATA FOR FOREGROUND PROGRAM OF KDP *
 ******************************************************
 INFILE
    DSNAME=TS0123.BFG.SAMPLE
    UNIT=3380
    VOLSER=DB6335

 PRINT
    COPY=1
    TITLE=YY
    FORMAT=FM87
    RANGE=ALL
    LINENO=A
    LET=Y
    OUTLINE=PRINT
    EDIT=(D:1:20 M:21:60)
    LUNAME=T1234567
    PAUSE=NN
    EJECT=Y
 * END OF DATA
 /*
 //
```

5.10 RSCS

The RSCS (Remote Spooling Communications Subsystem) is a special-purpose subsystem under the VM environment, which provides facilities to receive and transmit messages, files, commands, and jobs over a computer network. The RSCS is also used to send output to a terminal printer. It supports the DBCS printing function. DBCS data need to be enclosed with SO and SI. The field outlining (ruled line) function by the SA (Set Attribute) control character, whose format is the same as that of SCS (SNA Character String), is also supported.

The CP TAG command is used to specify the destination address of a spool file and keyword options for the terminal printer output. The TAG command syntax is:

```
TAG DEV vaddr destnode identifier (priority (options...*))

        * PRT=[APL|TEXT|NOTR|GRAPH|DBCS]
```

The following are detailed descriptions for the DBCS-related options of the PRT keyword.

DBCS specifies that the terminal printer has DBCS printing capability. RSCS will carry out the validity check of DBCS data.

Note: To specify the above options, it is necessary that the FEATURE parameter of RSCS START command is properly specified. For more information, refer to *RSCS Networking Version 2 Operation and Use.*

The following is a sample operation to print the file "EXAMPLE MEMO" on a terminal printer called "TERMPRT" , which is connected to a host computer called "JPNVM" . On this host computer, RSCS resides in a virtual machine that has the userid of "RSCSVM" .

```
CP  TAG DEV PRT JPNVM TERMPRT 50 PRT=DBCS
CP  SPOOL PRT TO RSCSVM
CMS PRINT EXAMPLE MEMO
```

5.11 Data Base Manager

5.11.1 DB2 and SQL/DS

DB2 (Database 2) and SQL/DS (Structured Query Language/Data System) are relational data base managers running under MVS (DB2) and VM/VSE (SQL/DS) environments. These products provide the functions which allow users to use DBCS user data, table names, column names, etc. The following functions are provided for DBCS users.

- Data types
  - CHAR
  - VARCHAR
  - LONG VARCHAR
  - GRAPHIC
  - VARGRAPHIC
  - LONG VARGRAPHIC

  The data types related to "CHAR" type are used to store Mixed strings as user data, and the data types related to "GRAPHIC" type are used to handle DBCS strings.

- Names

  Mixed strings can be used as the names of:

  - TABLE
  - COLUMN
  - VIEW
  - INDEX
  - SYNONYM
  - LABEL

- DBCS literal

  A DBCS literal can be used to assign a DBCS string as an initial value of a variable or a constant. The format of a DBCS literal is shown in Figure 21 on page 58.

DB2 and SQL/DS support DBCS similarly. The differences are shown in Figure 21 on page 58.

	DB2	SQL/DS
Mixed String Fix-up Function *	Supported	Not supported
TABLE/COLUMN Name Specification	Should be surrounded by escape characters.**	Escape character is not required.
DBCS Literal Specification	G'␎...␏'	G'␎....␏' or '␎....␏'
Collating Sequence (Ordering)	Supported ***	Not supported

Figure 21. Differences between DB2 and SQL/DS

Notes:

* This is the function to compensate SO/SI to Mixed strings when the unpaired SO/SIs are generated by truncation during the assignment operation.

** The escape character may be either quotation marks (") or apostrophes (') depending on the installation option.

*** The sequence value is obtained from an external routine that is incorporated into the DB2 system. The external routine is invoked through the FIELDPROC clause. DBCS Ordering Support Program, which is a IBM licensed program (program number : 5665-360), is usually used as the external routine for Japanese data.

Preprocessor (Translator)

Both DB2 and SQL/DS provide the preprocessors for programming languages such as COBOL or PL/I. Note that the DB2 preprocessor cannot handle DBCS data used as the variable names in source statements outside of SQL statements even if the language supports DBCS identifiers.

5.11.2 IMS/DB, DL/I

DBCS and Mixed data can be stored as CHARACTER data items that do not put the restriction on code range. DL/I does not care about the data contents. Applications should know the data type and have the coding for proper manipulation.

5.12 Languages

In this section, COBOL (VS COBOL II), PL/I (OS PL/I), FORTRAN (VS FORTRAN) and Assembler (ASSEMBLER H) are introduced as the programming languages that support DBCS data handling.

DBCS String Handling

DBCS data type: COBOL and PL/I provide the DBCS data type. Declare it as follows:

```
01   VAR1   PIC  G(10) USAGE DISPLAY-1.          (COBOL)

DCL  VAR2   GRAPHIC(10);                         (PL/I)
```

All the associated operations with the DBCS data type such as substring, concatenation, etc. are enabled to handle DBCS strings.

Note: "10" in the above specification denotes 10 DBCS characters, not 10 bytes.

DBCS literal: DBCS literals can be specified in the COBOL and PL/I source codes. The format of the DBCS literal is for example:

```
G"␎D B C S␏"        (COBOL)

'␎D B C S␏'G        (PL/I)
␎' D B C S ' G␏
```

DBCS literals can also be specified in Assembler source codes. The following shows an example of DBCS literals used in Assembler codes:

```
                                                Assigned Value
[label]   DC    G'␎D B C S␏'                      D B C S
[label]   DC    G'␎D B␏␎C S␏'                     D B C S
[label]   DC    GL4'␎D B C S␏'                    D B
[label]   DC    GL10'␎D B C S␏'                   D B C S ƀ
[label]   DC    3G'␎A␏'                           A A A
```

Note: Length attribute represents byte length, not DBCS character length.

Mixed String Handling

Mixed strings can be stored in Character data items supported by all four languages; COBOL, PL/I, FORTRAN, and Assembler. However, the support level of Mixed string handling is different among these languages.

Mixed literal: Mixed strings can be specified as literals/constants in source codes. The format of Mixed literals are syntactically expressed as character literals, except PL/I.

```
Example of Mixed Literal

  'sss␎D B C S␏sss'                  (FORTRAN)
  [label]   DC  C'sss␎D B C S␏sss'   (Assembler)
```

PL/I has two types of literals; literals expressed with and without the designator "M". The following shows the difference between two types of literals:

```
Example of Mixed Literal (PL/I)
                                         Assigned Value
  'abc␎D E F␏'                            abcDEF
  'abc␎D E F␏'M                           abc␎D E F␏
```

Mixed string compensation: Some languages provide the Mixed string compensation function, which make Mixed strings valid when unpaired SO/SI is generated during an assignment (move) or substring operation. Mixed strings can also be specified as macro operands in Assembler source codes.

- PL/I

  A new PROCEDURE and BEGIN block attribute is introduced to keep Mixed strings valid during any assignment operation. When the CHARGRAPHIC attribute is in effect for a procedure or a begin block as shown in the example below, the assignment operation of two strings that have different sizes from each other causes a library routine for Mixed string compensation to be called.

  ```
  name : PROCEDURE CHARGRAPHIC;

     DCL   A  CHAR(5);
     DCL   B  CHAR(8);

     A = B;
     /* ... will cause a library routine to be called. */

  END;
  ```

  PL/I also provides the built-in "MPSTR" function that is used for the assignment/substring operation to keep Mixed strings valid regardless of the CHARGRAPHIC attribute.

- FORTRAN

  FORTRAN provides the external routine ("ASSIGNM" routine) that assigns Mixed strings to other area keeping the validity of the Mixed string in case that the truncation of data occurs.

- Assembler

  Assembler provides no function used for Mixed string compensation at the execution time, however, it keeps the validity of Mixed strings given as literals/constants at the compilation time. The following results in a compiler error because an invalid Mixed string will be generated.

```
[label]   DC   CL3'␎ＤＢＣＳ␏'
```

Other functions: PL/I provides the following built-in functions for Mixed/DBCS string conversion.

GRAPHIC	converts a Mixed string to a DBCS string.
CHAR	converts a DBCS string to a Mixed string (DBCS string bracketed with SO/SI).

Compiler option

Each language has the compiler option (invocation option) for DBCS data handling. Users should specify the compiler option so that DBCS data is handled correctly.

GRAPHIC	(PL/I)
DBCS	(FORTRAN)
DBCS	(Assembler)

6.0 Message Translation Prerequisites

The total DBCS support is established together with National Language Support (NLS). This section does not cover the general NLS rules/guidelines[40] but covers only the items regarding DBCS.

The capability of DBCS presentation (both Full Screen Mode and Line Mode) is essential to NLS. In addition to this basic capability, there are some considerations required for national language support. The following are the main consideration items.

- Considerations on MRI (Machine Readable Information)[41]
  - MRI (Machine Readable Information) isolation
  - Panel Design
  - Message composition
  - Translatability
- Translation Management
  - Translation plan
  - Translation tool
- Multilingual Support
  - Language selection mechanism

6.1 Considerations on MRI (Machine Readable Information)

6.1.1 MRI (Machine Readable Information) Isolation

It is recommended that translated MRI (Machine Readable Information) be completely isolated from the executable code from the translatability and maintainability point of view. This means MRI should reside in a text repository that is separate from the executable code. There are two levels of MRI isolation to be considered.

[40] Refer to "National Language Information and Design Guide" (SE09-8001, SE09-8002) for the general NLS rules/guidelines. Also, "SAA Common User Access: Panel Design and User Interaction" (SC26-4358) contains useful information on DBCS panel design.

[41] MRI (Machine Readable Information) means all textual information contained in a program, which may appear on display panels or printers.

- Source level isolation (for translatability)

  MRI in the source code is not preferable for translation, because translators may not be familiar enough with programming to tailor MRI imbedded in the source code directly. The following is an example for isolation of MRI.

  ```
  (source code)
    PUT &text100
    If answer = &text200

  (text repository for MRI)
    100 Enter &text200 or &text210
    200 YES
    210 NO
  ```

  This is an example which is not suitable for translation since the MRI ("Enter YES or NO", "YES") is coded in-line.

  ```
  PUT ' Enter YES or NO '
  If answer = 'YES' ....
  ```

  Additionally, DBCS text in the macro is not preferable because the information (text, position, length, etc.) imbedded in the code needs to be modified for different languages, and the imbedded MRI scattered in the code may cause erroneous codes to be produced at the translation time and reduce productivity.

  ```
  (Example) CICS/BMS

    xxxxx    DFHMDF  POS=(n,m),
                     ATTRIB=(ASKIP,BRT),
                     INITIAL='......text........',
                     LENGTH=xx
  ```

- Object level isolation (for maintainability)

  Unless MRI is not object-level isolated, the update of MRI requires compilation and linkage each time. Besides, the object-level isolation of MRI enables users not to install unneeded MRI.

6.1.2 Panel Design

Sufficient space for MRI (Machine Readable Information): Since the length of the translated sentence is not equal to that of the original one, the expansion of the length should be allowable or the length restriction should be made known to the translators.

For example, a panel containing column-sensitive items may be designed so that multiple lines for column headings are allowed to accommodate sufficient space for translated column headings.

```
Orginal panel

   HEAD-1  HEAD-2  HEAD-3  ....
   ------  ------  ------
   XXXXXX  XXXXXX  XXXXXX

              |
              v

Translated panel

   TRANS   TRANS   TRANS
   HEAD-1  HEAD-2  HEAD-3  ....    <- Additional line
   ------  ------  ------
   XXXXXX  XXXXXX  XXXXXX
```

The expansion of MRI should not affect any function. In the following example, if the application assumes the exact position of each word, the position selection function will not work correctly when words are translated. Any function should not rely on the exact position of specific fields, columns, or words.

```
Original Panel

 1. HELP  2. SAVE  3. END  4. PREVIOUS  5. NEXT

Translated Panel

 1. ヘルプ  2. 保管  3. 終了  4. 前画面  5. 次画面

                  ^        ^        ^        ^
                  |________|________|________|
                     column position changed
```

6.1.3 Message Composition

Generally messages should be complete entities and not be constituted from individual words at the execution time from the translatability point of view. In case that a message is constituted from individual words or phrases, "NOUN" is only allowed in the substitution process. Others like "VERB", "PREPOSITION" cannot be inserted. A word should be used with a unique meaning. Do not use the same word in different meanings.

```
(Example)
            'plant' is in repository to be translated
          1. English    Take care of 'plant'.
             Japanese   .............'␎KJ-vegetable␏'
          2. English    'Plant'  export ....
             Japanese   '␎KJ-factory␏' ....

   A different word should be used in the second English sentence.
   There is no word in Japanese that has both meanings, vegetable
   and factory.
```

The substitution/imbedding of variables into a message needs the logic of Mixed string handling such as adjacent SO/SI removal, truncation, etc.

```
(example)  Orginal message =␎D1D2␏&1.␎D4D5␏

           &1. = abcd       Result:               ␎D1D2␏abcd␎D4D5␏

                            Appearance:            D1D2 abcd D4D5

           &1. = ␎D6D7␏     Intermediate result: ␎D1D2␏␎D6D7␏␎D4D5␏
                                                      ──    ──
                                                remove adjacent SO/SI

                             Final result:         ␎D1D2D6D7D4D5␏

                             Appearance:            D1D2D6D7D4D5
```

6.1.4 Translatability

There is no DBCS unique consideration from the translatability point of view except for mnemonic selection. Some applications allow users to select some action using a mnemonic character from an interactive panel as follows:

```
Action _  :  Help   Execute  End  Print
```

When these applications are translated into DBCS languages, the SBCS mnemonic character should be kept because DBCS mnemonic characters are not appropriate. Therefore, applications should take it into consideration that MRI (Machine Readable Information) allows translators to easily find which character in each text is used as a mnemonic character, as well as that the sufficient spaces are accommodated for the translation.

Original panel

```
Action _  :  Help        Execute        End           Print
```

Translated panel

```
アクション  _ : ヘルプ (H)   実行 (X)   終了 (E)   印刷 (P)
```

6.2 Translation Management

It is essential to keep the translation consistent among the base code, MRI and the documents through different version or releases. To keep translation consistent, a tool to generate a cross reference for the used words may be required.

An application may have its own translation tool or simulation tool that is used to validate the complete messages that users will see. In such a case, the associated tools should also be enabled to handle DBCS data properly.

```
:xample)

      English   message    msg.id  sss..ss &i sss &j sss....s
                                   &i = 'xxxx'
                                   &j = '123'
                             |
                       translation
                             ↓
      Japanese  message    msg.id  ␎D1..D2␏ &j ␎D3␏ &i ␎D4....D5␏
                                   &i = '␎Di␏'
                                   &j = '123'
                             |
                      Simulation tool
                             ↓
                          msg.id  ␎D1..D2␏123␎D3DiD4....D5␏ (Complete message)
```

6.3 Multilingual Support

Multilingual support, which is usually the support of bilingual messages/panels, may be required to some applications. This is because the SBCS and DBCS terminals are often connected to the same Host application. Therefore, such applications that are executed under both SBCS and DBCS environments should provide an SBCS (usually English) message module for SBCS terminals and a DBCS (Japanese, Korean, or Chinese) message module for DBCS terminals. It is not required to provide multiple DBCS message modules for a single DBCS terminal. The following are items related to the multilingual support.

Language selection: There are several approaches for selecting a language to be used in the system depending on the operation modes (batch or interactive). If the application runs in the batch mode, it provides language selection capability by an install selection (install base) or a user selection (job base by exec parameter). If the application runs in the interactive mode, it must also provide language selection capability by an install selection or by a user selection such as the user profile or the specification during the session.

In the interactive mode, there sometimes arises conflict between the selected language and the character set supported by the terminal. To avoid this, such a function is required that determines an adequate language by checking if the terminal supports DBCS.

For Japanese, even if all of the terminals are SBCS, applications must support both U.S-English terminals and Japanese-Katakana terminals at the same time.[42] See 6.4, "Katakana Terminal Support" below.

Note: For a general guideline for terminal recognition, see the description of the terminal recognition in the appendix, which shows how to recognize terminal characteristics under a variety of environments. For the 3270 data stream approach, see the description of A.1.3, "Query Reply" on page 79.

Language selection ID: When the application provides a language selection mechanism, IBM recommends the following keywords as the language selection ID.

	Korean	Japanese	Simplified Chinese	Traditional Chinese	Thai
3 bytes	KOR	JPN	CHS	CHT	THA
8 bytes	KOREAN	JAPANESE	CHINESES	CHINISET	THAI
Full (*)	Korean	Japanese	Simplified Chinese	Traditional Chinese	Thai

Figure 22. Language Section ID

Note: It is recommended that an ID should be eight bytes long unless any length limitation exists.
(*) Full is used in descriptive text.

6.4 Katakana Terminal Support

In the Japanese environments, Katakana terminal support[43] is required because many terminals are installed with the Katakana feature, where single-byte lowercase alphabetic characters are displayed as single-byte Katakana characters.[44] When an application with English message/panel texts runs on a Katakana terminal, lowercase alphabetic characters in the text are displayed as meaningless Katakana.

[42] VTAM V3.1.1 with PTF supports an additional one bit (LANG parameter) in VTAM LOGMODE table which is used to indicate whether or not the terminal supports Japanese-Katakana character set. TSO will support the GTTERM macro which returns the terminal characteristics such as DBCS character set ID and SBCS character set ID (Japanese-Katakana or U.S-English).

[43] Katakana terminal support described in this section is independent of DBCS support, but a Japan-unique problem that applications running in the Japanese environments should take into consideration.

[44] Katakana codes are assigned to the same code points as lowercase alphabetic characters. This means that Katakana and lowercase alphabetic characters are mutually exclusive.

To solve the problem, one of the following approaches is normally taken.

- All MRI is provided in uppercase English only.
- MRI is provided in both mixed case and uppercase versions which are selected based on the terminal type.
- Before sending panels/messages to a terminal, the product converts lowercase alphabetic characters to uppercase depending on the terminal type.

Appendix A. Device Data Stream

When presenting DBCS data on displays or terminal printers, two kinds of device data streams are used. One is the 3270 extended data stream (3270EDS)[45], the other is the SNA Character String (SCS).

3270EDS is used as the data stream for LU-2 displays and for LU-3/non-SNA printers. SCS is used as the data stream for LU-1 printers.

A.1 3270 Extended Data Stream

The 3270 extended data stream provides the capability to define additional properties of a field beyond those defined by the base 3270 data stream. These properties are, for example, SO/SI creation, field outlining, or extended high-lighting. These are called "Extended Attributes". The extended attributes cannot be defined on all types of terminals, so you must check if the terminal can accept extended data stream before sending it.

This chapter describes the following;

- How to define extended attributes.
- How to recognize the terminal characteristics.

A.1.1 Extended Attribute

The 3270 extended data stream provides the following three kinds of attributes:

Field Attribute	defines the start of a field and provides the basic properties, such as display/non-display, protected/unprotected, or etc.
Extended Field Attribute	provides additional properties of a field to those provided by field attribute. The extended field attribute is always associated with a field attribute. These attributes do not occupy positions in the character buffer.

[45] 3270 Extended Data Stream is the device data stream which has added the capability of extended field and character attributes to the 3270 Data Stream.

Refer to *IBM 3270 Information Display System Data Stream Programmer's Reference* (GA23-0059) for general information on 3270 data stream.

Character Attribute	controls the characteristics of an individual character. Character attributes do not occupy positions in the character buffer. The extended field attributes of any single character are superseded by the character attributes associated with it.

The figure below shows a conceptual view of the extended field and character attribute.

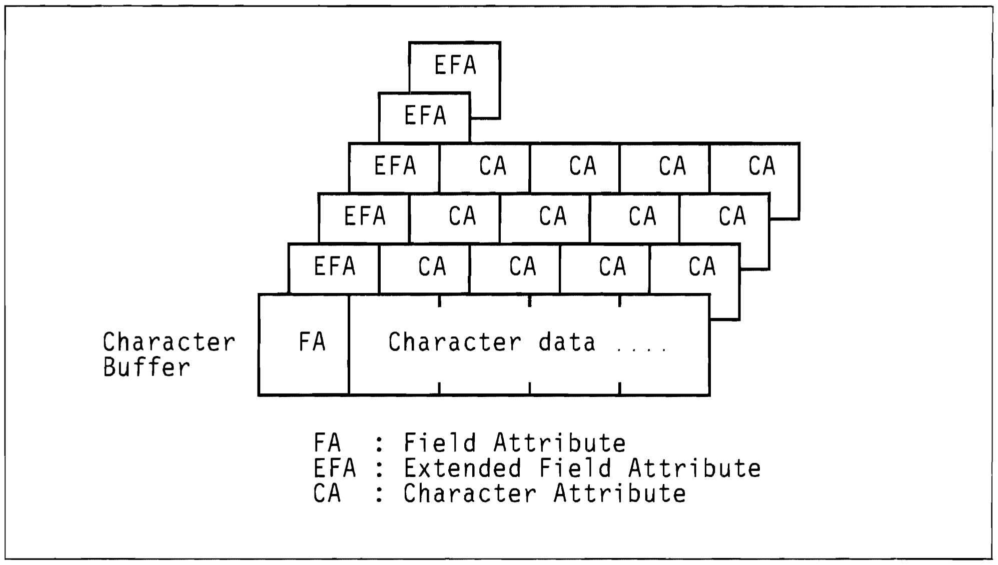


Figure 23. Conceptual View of Extended Field and Character Attribute

Extended Field Attribute: On the terminals which can accept the extended data stream, the properties of each field are defined by the combination of one basic attribute and some extended field attributes. The basic attribute provides the same properties as those that can be defined on the current 3270 terminals. And the extended field attribute provides the following properties.

Property	Attr. Type	Attr. Value	Description
Field Attribute (3270 Field Attribute)	X'C0'	X'00' X'nn'	Default Field Attribute Value

Figure 24 (Part 1 of 2). Extended Field Attributes

Property	Attr. Type	Attr. Value	Description
Extended Highlighting	X'41'	X'00' X'F1' X'F2' X'F4'	Default Blink Reversed video Underscore
Extended Color	X'42'	X'00' X'F1'-X'F7'	Default Set seven different colors
Character Set *	X'43'	X'00' X'F8'	Default character set (SBCS) Primary Double-Byte Character Set
Transparency	X'46'	X'00' X'FF'	Default No transparency
Field Outlining	X'C2'	X'00'-X'0F'	Set field outlining, 16 combinations can be set **
SO/SI creation ***	X'FE'	X'01' X'00'	SO/SI creation enable SO/SI creation disable

Figure 24 (Part 2 of 2). Extended Field Attributes

* When a DBCS field is defined, this type is used and followed by the value (X'F8').

** When the field is defined to have field outlining, this type is used. The value is as follows:

Bit	Value	Meaning
0-3	B'0000'	Reserved
4	B'1'	If on, draw left line
5	B'1'	If on, draw over line
6	B'1'	If on, draw right line
7	B'1'	If on, draw under line

When bits 4 - 7 are all on, it designates a BOX.

*** When a field is defined as "SO/SI is allowed to be input by an operator," this type is used.

You can specify extended field attributes by using the SFE (Start Field Extended) order or MF (Modify Field) order. Each extended attribute is specified by the pair of its type and value.

The format of the SFE/MF order is:

Order	Number of Attr.pair	type	value	...	Data ...

```
   SFE (X'29')
or MF  (X'2C')
```

__Sample Data Stream__

1. Define DBCS input field

Byte	Value (Hex)	Meaning
0	F1	Write command
1	C3	WCC
2	11	SBA
3,4	aaaa	Field address
5	29	SFE
6	02	Number of pairs
7	C0	Field attribute
8	40	value
9	43	Character set control
10	F8	value (set DBCS)

2. Define Mixed input field with field outlining

Byte	Value (Hex)	Meaning
0	F1	Write command
1	C3	WCC
2	11	SBA
3,4	aaaa	Field address
5	29	SFE
6	03	Number of pairs
7	C0	Field attribute
8	40	value
9	FE	SO/SI creation control
10	01	value
11	C2	Outlining control
12	0F	value

Note: If the field attribute (X'C0') is not specified, it is set to the default value (X'00').

Character Attribute: Character attributes are associated with an individual character to define the characteristics of the character. The character attribute provides the following characteristics:

- Extended highlighting (attribute type = X'41')
- Extended color (attribute type = X'42')
- Character set (attribute type = X'43')
- Transparency (attribute type = X'46')[46]

[46] The transparency attribute can be specified for graphic terminals only.

You can specify character attributes by using the SA (Set Attribute) order. Similar to extended field attributes, character attributes are specified by the combination of attribute type and value. (Refer to "Extended Field Attribute")

The format of the SA order is:

SA (X'28')	Attr. type	Attr. value	Data ...

The characteristics defined by the SA order is effective until;

- A new SA order changes it.
- Another write-type command is sent.
- CLEAR key is pressed.
- Power at the display is switched off.

The character attribute occupies no screen position. So you can present various data which have different properties without any gaps (spaces) between the data. For example, you can present Mixed string without space between the DBCS and SBCS portions.

Sample Data Stream

1. Output of Mixed data without SO/SI as follows;

   ```
   eeeD1D2D3eee
   ```

Byte	Value (Hex)	Meaning
0	F1	Write command
1	C3	WCC
2	11	SBA
3,4	aaaa	Field address
5	1D	SF
6	60	Field attribute
7-9	eee	(represented character image
10	28	SA
11	43	Character set control
12	F8	value (set DBCS)
13-18	D1D2D3	(represented character image
19	28	SA

Byte	**Value (Hex)**	**Meaning**
20	43	Character set control
21	0F	value (reset)
22-24	eee	(represented character image

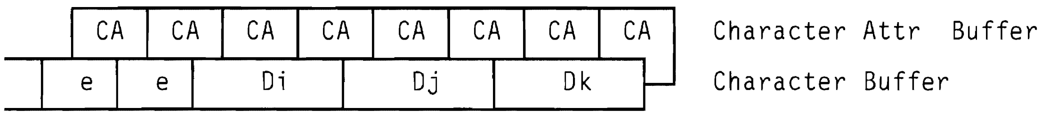
Special consideration for Mixed data with character attribute: When doing an I/O operation of Mixed data with character attributes, some special considerations are required, especially when reading it from a terminal.

- Reply mode

  An application program can only get the data in the character buffer by a READ operation under the usual reply mode[47] (Field mode) of the terminal. Then there is no delimiter between SBCS and DBCS characters in the data because the character attributes used to identify DBCS and SBCS are put in the character attribute buffer. In order to avoid such a situation, the application programs must set the reply mode to Character mode.[48] Under this mode, character attributes are converted to SA orders during any READ operation.

  <u>**Example**</u>

  When Mixed data is presented as follows;



  1. Under "Field mode"

     The string "eeDiDjDk" is returned to the Host by any READ operation.

  2. Under "Character mode"

     The string "eeSA1DiDjDkSA2" is returned.
     where SA1 = X'2843F8'
     SA2 = X'284300'

  Then by replacing SA1/SA2 with SO/SI respectively, an application program can handle Mixed data normally.

- Length of string

  When an application program has read a Mixed string with character attributes from a display and converted SA orders to SO/SI characters as

[47] Reply mode is a mode which defines the format of an inbound data stream generated in response of READ commands. Application programs can specify the reply mode through the Set Reply Mode structured field, which is introduced with Write Structured Field (WSF) command.

[48] Send X'F3'(WSF) + X'000609000243' to set the reply mode to Character mode.

described above, it must pay attention that the length of the Mixed string is longer than that of field defined on the display.

```
On display        In storage

sssD1D2D3         sss␎D1D2D3␏

  9 bytes          11 bytes
```

The application program should prepare the data item which is longer in length than the field, or which has varying length in order to do I/O operations properly.

A.1.2 Orders and 3270EDS Unique Characters

1. Orders

In 3270 Extended Data Stream, the same orders as 3270 Data Stream are used. However, some considerations are needed when presenting DBCS data.

- RA order (X'3C')

  An RA order can be used to repeat a DBCS character with the following format.

```
Byte  ┌──────────────┐
 0    │ RA Order     │
      ├──────────────┤
 1    │ Stop         │  Should not point to the 2nd byte of
 2    │   Address    │  a DBCS character if this writes over
      ├──────────────┤  a DBCS field or a DBCS portion.
 3    │ Character to │
 4    │  be repeated │
      └──────────────┘
```

Note: A DBCS RA order is five bytes long while an EBCDIC one is four bytes long.

```
Example
            Data :  ␎...D14040....4040␏
              DBCS RA =  ␎...D1RAxxxx4040␏     Correct
                      =  ␎...D1RAxxxx40␏       Error
```

- EUA order (X'12')

  An EUA order can be used to insert nulls in an unprotected DBCS field or portion with the following format:

Byte		
0	EUA order	
1 2	Stop Address	Should not point to the 2nd byte of a DBCS character if this writes over a DBCS field or DBCS portion.

- SBA order (X'11')

  An SBA order can be used to set the current buffer address to within a DBCS field or portion with the following format:

Byte		
0	SBA order	
1 2	Stop Address	Should not point to the 2nd byte of a DBCS character.

- IC order (X'13')

  There is no restriction about the address but if a program tries to set the cursor to the second byte of a DBCS character, the terminal automatically sets it to the first byte.

2. ***3270 Extended Data Stream Unique Control Character***

- 3270 Special Device Controls

  The following two-byte controls are defined for printers. They are used by printer support programs and are generally not needed by application programs.

X'0000'	...	null
X'000C'	...	Forms Feed
X'000D'	...	Carriage Return
X'0015'	...	New Line
X'0019'	...	End of Message
X'001C'	...	Duplicate
X'001E'	...	Field Mark

  Note: On the display and the printer
  X'001C' is displayed as *
  X'001E' is displayed as ;

- Designator Characters for Cursor Select Keys and Mouse

  The following two byte designator characters are supported by 5550.

X'0000'	...	Attention
X'4040'	...	Attention
X'4250'	...	Enter simulation
X'426E'	...	Field selected
X'426F'	...	Field not selected

- For output, only a Mixed string can also be transmitted outbound by using an SF order. This is an output only hardware function.

A.1.3 Query Reply

Before presenting DBCS data on a terminal, application programs should determine whether the terminal has DBCS presentation capability, because terminals without this capability may hang if the application sends DBCS data.

The information on the terminal characteristics can be obtained from presentation service programs such as ISPF or GDDM, or obtained through "Query Reply Structured Field" .

Terminal Query by Presentation Service Program: When using presentation service programs, application programs can easily get the terminal characteristics. The following shows the presentation service mechanism for terminal query.

- ISPF

  If an application program checks the variable ZDBCS and finds it set to a value of YES, (ZDBCS=YES), then the program knows that DBCS exists on the workstation.

- CICS/VS

  Use the command EXEC ASSIGN to find the terminal characteristics.

- VSE/ICCF

  Use the macro DTSSCRN to find the terminal characteristics.

- GDDM

  Use the call FSQURY to find the terminal characteristics.

Terminal Query by Application programs: When application programs do not use any presentation service programs, they can get the terminal characteristics through structured fields of the terminal.[49] System macro/instruction are provided so that they can easily query the terminals. The following is a sample coding.

[49] Write Structured Field (WSF) command of the 3270 data stream is used to issue the query request.

1. TSO

```
                 TPUT  RESET,RESETLEN,NOEDIT
                 TPG   QUERY,QUERYLEN,NOEDIT,,HOLD
                 TGET  RECV,RECVLEN
                   :
        RESET    DC    X'F13C'
        RESETLEN EQU   *-RESET
        QUERY    DC    X'F3000501FF02'
        QUERYLEN EQU   *-QUERY
        RECV     DS    nnF
        RECVLEN  EQU   *-RECV
```

When VTAM 3.1.1 with PTF[50] or the later release is installed in your system, you can query the terminal characteristics by using the GTTERM macro as follows.

```
                 LA    Rx,PRMS
                 LA    Ry,ATTR
                 GTTERM PRMSZE=Rx,ATTRIB=Ry
                       :
        PRMS     DS    CL2
        ATTR     DS    F
```

Then the GTTERM macro returns the information on the terminal attributes into the "ATTR" field. The contents of this field are:

Byte	**Bit**	**Value**	**Meaning**
0			Reserved
1	0	0	Terminal does not support DBCS
		1	Terminal supports DBCS
	1-7	0000000	US English (default)
		0000001	US English
		0010001	Katakana
2			Reserved
3	0-5		Reserved
	6	0	Terminal supports EBCDIC code
		1	Terminal supports ASCII code
	7	0	Terminal does not support Query
		1	Terminal supports Query

2. CMS

```
                LA    Rx,INBUF
                LA    Rx+1,9
                LA    Ry,length of buffer
                DIAG  Rx,Ry,(X'8C')
                  :
        INBUF   DS    nnF
```

[50] PTF Number : UY90030 on 5665-289 (VTAM 3.1.1. for MVS/XA)
PTF Number : UY90031 on 5665-313 (VTAM 3.1.1. for MVS)

Then the DIAGNOSE instruction with code X'8C' returns the Query Reply structured Fields into the "INBUF" buffer.

Query Reply Structured Field

The information, which an application program can get through the TGET macro (TSO) or DIAG instruction (CMS), consists of a series of various structured fields (Query Reply Structured Field). The application program should extract the structured fields related to DBCS capability from it.

The DBCS related Query Reply Structured Fields, which are implemented under DBCS workstations, are discussed here. They are "DBCS-Asia Query Reply Structured Field," "Character Set Query Reply Structured Field," and "Field Outlining Query Reply Structured Field" .

1. DBCS-Asia Query Reply Structured Field

   Function

   This query reply structured field indicates that the shift out character set is DBCS and creation of SO/SI by the operator is supported.

   Format

Byte	Bit	Contents	Meaning
0-1		X'000B'	Length of structured field
2		X'81'	Query reply identifier
3		***X'91'***	***Identifies this reply as input control***
4		X'00'	Flag (reserved)
5		X'03'	Length of self defining parameter
6		X'01'	SO/SI self defining parameter
7		X'80'	Set ID of the shift out character set
8		X'03'	Length of self defining parameter
9		X'02'	Input control self defining parameter
10	0-6 *7*	B'000000' ***B'1'***	Reserved ***SO/SI creation supported***

Figure 25. DBCS-Asia Query Reply Structured Field

   Note: The combination of byte 3 and bit 7 of byte 10 indicates that the workstation has a DBCS capability.

2. Character Set Query Reply Structured Field

Function

This query reply structured field indicates the number and the kind of character set.

Format

Byte	Bit	Contents	Meaning
0-1		X'0023'	Length of structured field
2		X'81'	Query reply identifier
3		***X'85'***	***Identifies this reply as character set***
4	0 1 2 3 4 5 6 7	B'1' B'0' B'0' B'0' B'1' B'1' B'1' B'0'	Graphic Escape recognized Reserved Load Programmed Symbols structured field not supported Load Programmed Symbols structured field extension not supported Variable size matrix Two-byte coded character set supported GCSGID present Reserved
5		X'00'	Reserved
6		X'nn'	Default dot matrix block width X'08' 16 × 16 Display or Printer X'0C' 24 × 24 Display or Printer
7		X'nn'	Default dot matrix block height X'10' 16 × 16 Display or Printer X'18' 24 × 24 Display or Printer
8-11		X'00000000'	FORM : N/A
12		X'0B'	Length of descriptors which follow
13-n			Descriptor

Figure 26. Character Set Query Reply Structured Field

Descriptors: (There are two entries)

Entry-1: For one-byte character set

Byte	Bit	Contents	Meaning
0		X'00'	Device specific character set ID
1	0 1 2 3 4-7	B'0' B'0' B'0' B'0' B'0000'	Non-loadable character set Single plane character set One-byte coded character set LCID compare Reserved

Figure 27 (Part 1 of 2). Entry-1 in Character Set Query Reply Structured Field

Byte	Bit	Contents	Meaning
2		X′00′	Local character set ID (Alias)
3		X′00′	Character set cell width = Default (byte 6)
4		X′00′	Character set cell height = Default (byte 7)
5		X′00′	Starting subsection: N/A
6		X′00′	Ending subsection: N/A
7-10		X′nnnnmmmm′	CGCSGID for one-byte character set X′014C0122′ for Japanese Katakana X′00650025′ for US English X′03A50341′ for Korean one-byte X′03A80344′ for Simpl. Chinese one-byte

Figure 27 (Part 2 of 2). Entry-1 in Character Set Query Reply Structured Field

Note: In the case of Japan, both CGCSGID′s ("Japanese Katakana" and "U.S English") are returned.

Entry-2: For DBCS

Byte	Bit	Contents	Meaning
0		X′80′	Device specific character set ID
1	0 1 2 3 4-7	B′0′ B′0′ B′1′ B′0′ B′0000′	Non-loadable character set Single plane character set Two-byte coded character set LCID compare Reserved
2		X′F8′	Local character set ID (Alias)
3		X′nn′	Character set sell width X′10′ 16 × 16 Display or Printer X′18′ 24 × 24 Display or Printer
4		X′nn′	Character set sell height X′10′ 16 × 16 Display or Printer X′18′ 24 × 24 Display or Printer
5		X′41′	Starting subsection
6		X′00′	Ending subsection X′7F′ Japanese DBCS X′FE′ Korea, ROC, PRC DBCS
7-10		X′nnnnmmmm′	CGCSGID for DBCS X′0172012C′ for Japanese X′03A60342′ for Korean X′03A70343′ for Traditional Chinese X′03A90345′ for Simplified Chinese

Figure 28. Entry-2 in Character Set Query Reply Structured Field

Note: The LCID and CGCSGID associated with the Set ID of the shift out character set is reported in the "Character Set Query Reply Structured Field" . DBCS workstations implement that DBCS character set for Japan/Asian countries and have a common Set ID (X'80') and LCID (X'F8').

3. Field Outlining Query Reply Structured Field

Function

This query reply structured field specifies the details of the field outlining supported by the device.

Format

Byte	Bit	Contents	Meaning
0-1		X'000A'	Length of structured field
2		X'81'	Query reply identifier
3		***X'8C'***	***Identifies this reply as field outlining***
4		X'00'	Flag (Reserved)
5		X'00'	Underline/Overline separation: NO
6		X'nn'	Location of vertical line for display For example; X'00' Leftmost X'04' 5th dot X'06' 7th dot
7		X'00'	Location of Overline/Underline: Above cell
8		X'00'	Location of Overline: N/A
9		X'00'	Location of Underline: N/A

Figure 29. Field Outlining Query Reply Structured Field

Note: Byte 3 indicates that the workstation has a Field Outlining capability.

A.1.4 SNA Sense Code

The following are the DBCS unique SNA sense codes and the causes of errors.

- X'1003' (Unsupported function)
  - The attribute type of SFE or MF order is X'43' (Character Set), and its attribute type is X'FF'.
  - The attribute type of SFE or MF is X'C2' (Field outlining), however bits 0-4 of the attribute value are not zero.
  - The attribute type of SFE or MF is X'FE' (SO/SI creation), however bits 0-6 of the attribute value are not zero.

  - The outbound data stream to the DBCS field contains SO/SI.
  - Shift Out is sent to the DBCS portion.
  - SO/SI is not pairing.

- X'1005' (Parameter error)

  - The first byte of a DBCS character does not locate at an odd boundary of the DBCS portion or DBCS field.
  - The parameter of an RA or EUA order indicates the current buffer address or the stop address which locates at an even boundary of a DBCS field or DBCS portion.

- X'0863' (Invalid character set)

  - The attribute type of SFE order is X'43', however its attribute type is not X'00', X'F8', or X'FF'.
    (When the attribute type is X'FF', sense code X'1103'
    is sent.)

Refer to *IBM 3270 Information Display System Data Stream Programmer's Reference* for other sense codes.

A.2 SNA Character String

The SNA Character String (SCS) is used as the data stream for LU-1 printers. It consists of the following components:

1. SCS Control Character
2. Data
3. FMH1 (Function Management Header1)
4. Structured field

Note: FMH1 is used as a header of a structured field. The structured field is used to the read partition structured field that shows the characteristics of printers.

A.2.1 SCS Control Character

The SCS control characters exist in an outbound data stream to printers. The following control characters should be considered to present DBCS data.

Shift-Out (SO)	Shift-Out control character (X'0E') indicates the beginning of a DBCS portion in data. SO does not occupy a print position.
Shift-In (SI)	Shift-In control character (X'0F') indicates the beginning of a SBCS portion. SI does not occupy a print position.

SO/SI control characters are used for character set switching between SBCS and DBCS.

Set Attribute (SA)	Set Attribute control character (X'28') is used for character set switching and for grid lines. The format of the SA control character is same as the Set Attribute order of 3270 Extended Data Stream.

X'28'	Attr.Type	Attr.Value	Data ...

Property	Attr. Type	Attr. Value	Description
Character set	X'43'	X'00' X'F8'	Beginning of SBCS portion. Beginning of DBCS portion.
Grid line	X'C2'	X'00'-X'0F'	Set grid line. 16 different grid lines can be specified. *
Attribute reset	X'00'	X'00'	Set all attributes to default value. = X'4300' and X'C200'

Figure 30. Characteristics Defined by SA Control Character

* The values are as follows:

Bit	Value	Meaning
0-3	B'0000'	Reserved
4	B'1'	If one, draw left line
5	B'1'	If one, draw over line
6	B'1'	If one, draw right line
7	B'1'	If one, draw under line

Note: When designating grid lines, the following considerations are necessary.

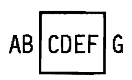
1. The SA control character to designate grid lines should not be put in the DBCS portion (between SO and SI).

The following data string is an erroneous data stream.

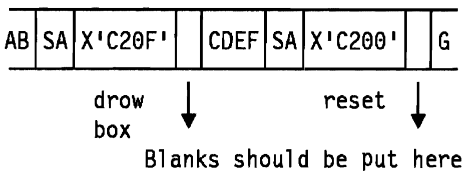
SO	DBCS data	SA	X'C20F'	DBCS data ...	SI

2. When designating vertical grid lines, it is better to put blanks following the SA control character, because vertical grid lines are printed over the character that follows the SA control character.

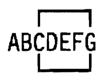
Designate grid lines as follows;



you should create SCS as follows;



If you don't include blanks, the data is printed as follows;



Appendix B. Advanced Presentation

B.1 Boundary Management

For advanced display operations such as windowing, it is necessary to manage the data display boundaries. In this chapter, the basic rules required in the following cases are described.

- Scrolling
- Partitioning
- Wrapping

Generally the rules described here are implemented by presentation service programs. Applications that provide their own display I/O support need to consider these rules.

B.1.1 Scrolling

Scrolling is mainly required by EDIT and BROWSE type functions. Scrolling allows the user to move the screen "window" left, right, up, or down across the information. During EDIT and BROWSE, the user might deal with information that exceeds the screen size or field size.

In contrast to the SBCS character, applications find many boundary handling requirements of DBCS characters. When the scrolling area starts or ends in the middle of a DBCS portion of a Mixed string, some compensation is required to make the string meaningful.

There are two reasons for compensation. The first is to show a Mixed string properly for displaying, and the second is to maintain the original string properly from user's updating for integrity.

Left/Right Scrolling

DBCS string: The scrolling amount should be an even number of bytes in length, exclusive of attribute bytes.

Mixed string: When scrolling Mixed data, a DBCS portion can cross the left screen edge, the right screen edge, or both left and right screen edges. Then the compensation of unpaired SO/SIs is required.

```
    Original String        Screen View

         LEFT               LEFT
       BOUNDARY           BOUNDARY

          |                  |
       eeee␎D1D2D3␏eee      e␎D1D2D3␏eee   No compensation
Scroll    |                  |
  3 Left  |                  |
    eeee␎D1D2D3␏eee          ␎D2D3␏eee      Compensation
```

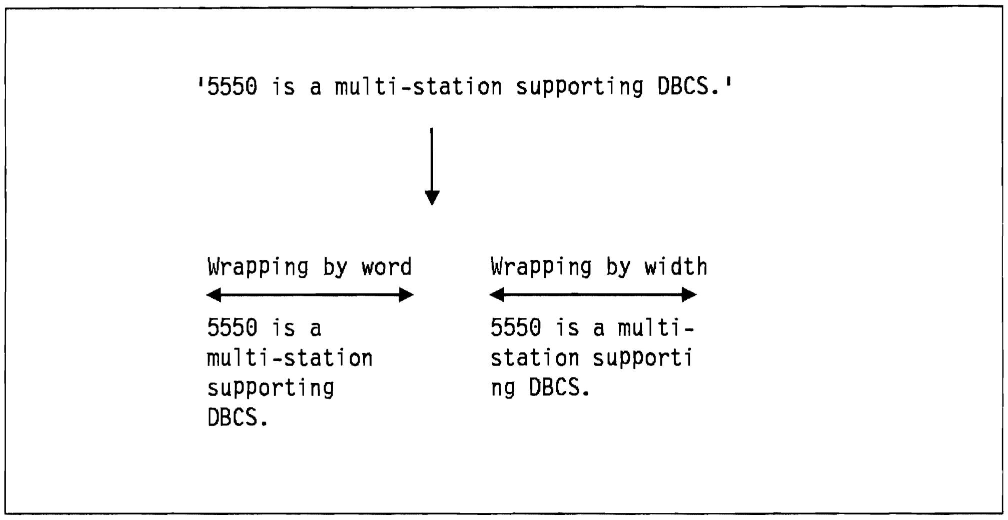
Up/Down Scrolling

Same considerations as shown in left/right scrolling are needed when a DBCS character/portion crosses the left/right edge after an up/down scrolling. In addition, if the window is defined as one physical field, the same considerations are also needed at both field ends because the first or last line may contain a DBCS string which continues outside the window.

B.1.2 Text Wrapping

A Text wrapping operation can be categorized into two types; Wrapping by word, and Wrapping by width. Word wrapping is not applicable to most DBCS text (especially Japanese) because Japanese words are not separated by delimiters. As for wrapping by width, different rules apply to DBCS strings and Mixed strings.

Wrapping by width is required when dealing with DBCS strings. Wrapping by a combination of words and width is required when dealing with Mixed strings.



DBCS String Wrapping

Wrapping by Width: The width applied to a DBCS field counts each double byte character as two bytes.

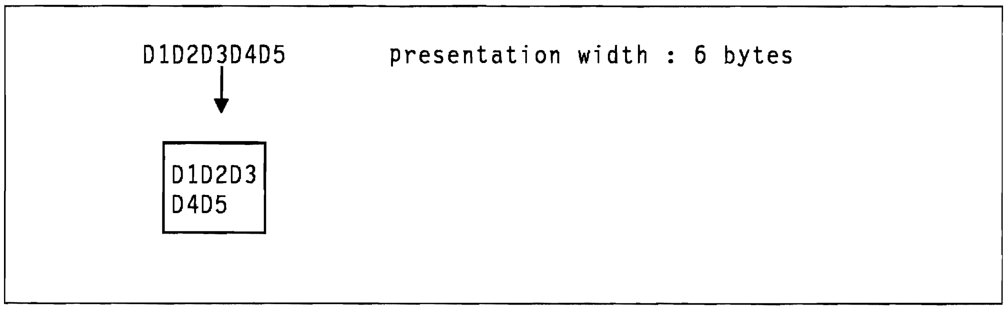


Figure 31. Example of Wrapping by Width of a DBCS String

Mixed String Wrapping

Wrapping by width: The length applied to a Mixed field is counted in bytes. A DBCS character cannot be split.

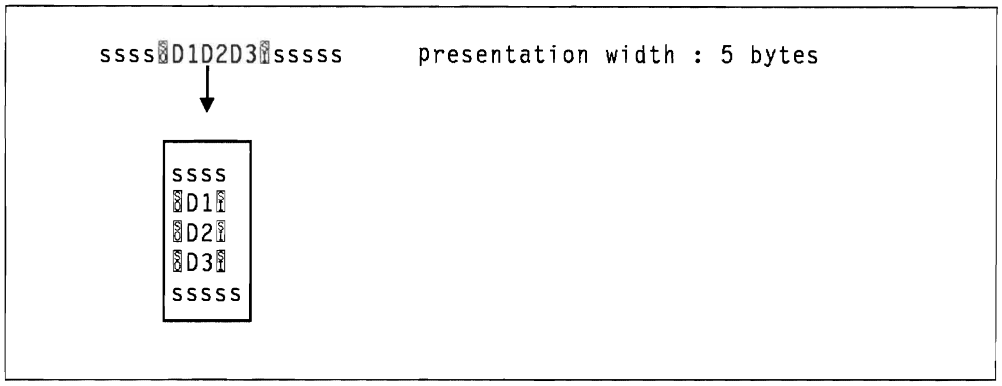


Figure 32. Example of Wrapping by Width

Wrapping by Word: A DBCS space is handled the same as a SBCS space.

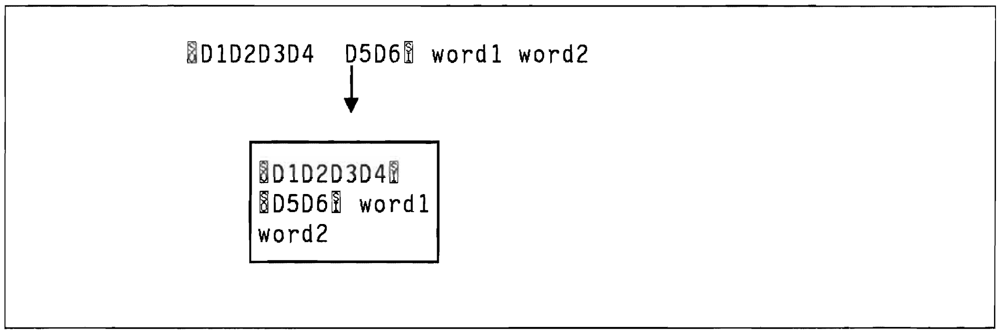


Figure 33. Example of Wrapping by Word

Wrapping Combined word and width: English text can be wrapped by word but Japanese text cannot. When both English text and Japanese text are Mixed, wrapping by word can be applied to only the English text like:

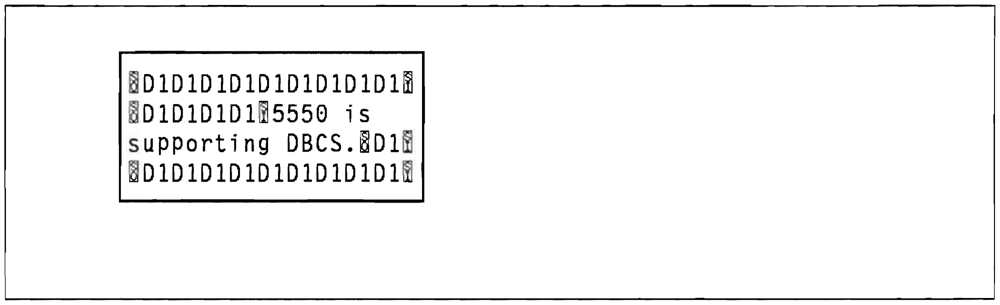


Figure 34. Example of Wrapping Combined Word and Width

Note: Wrapping by width was applied to the DBCS portion. However, text processing requires other linguistic considerations. For example, a Japanese period cannot occupy the first position of a new line.

B.1.3 Windowing

Partitioning is normally done by overriding a portion of an already displayed area. If a DBCS field or a DBCS portion is divided by the partition boundary, extra care is required in addition to the basic considerations associated with Scrolling.

Basic (SBCS field) consideration: If a field is divided into two fields, the attribute of a divided field should be carried over to the second portion of the divided field. This is a very simple consideration if you think about a green color field, a part of which is overridden by a red field. If the green attribute is not carried over, the remaining portion of the green field displays as red.

DBCS field consideration: If a DBCS field is divided, any partial fields should be adjusted so that they are an even number of bytes in length (excluding any attribute bytes). If they are not an even number of bytes in length, a presentation service program must make them an even number of bytes in length by adding an attribute byte to the start of the field or at the end of the field.

Mixed field consideration: If a Mixed field is divided at an SBCS portion, no special consideration is required. If it is divided at a DBCS portion, a presentation service program must make the remaining DBCS portion a valid DBCS portion by adding SO and/or SI appropriately. The length of the DBCS portion should be made an even number of bytes and should be enclosed with SO and SI characters at the appropriate screen or partition boundary.

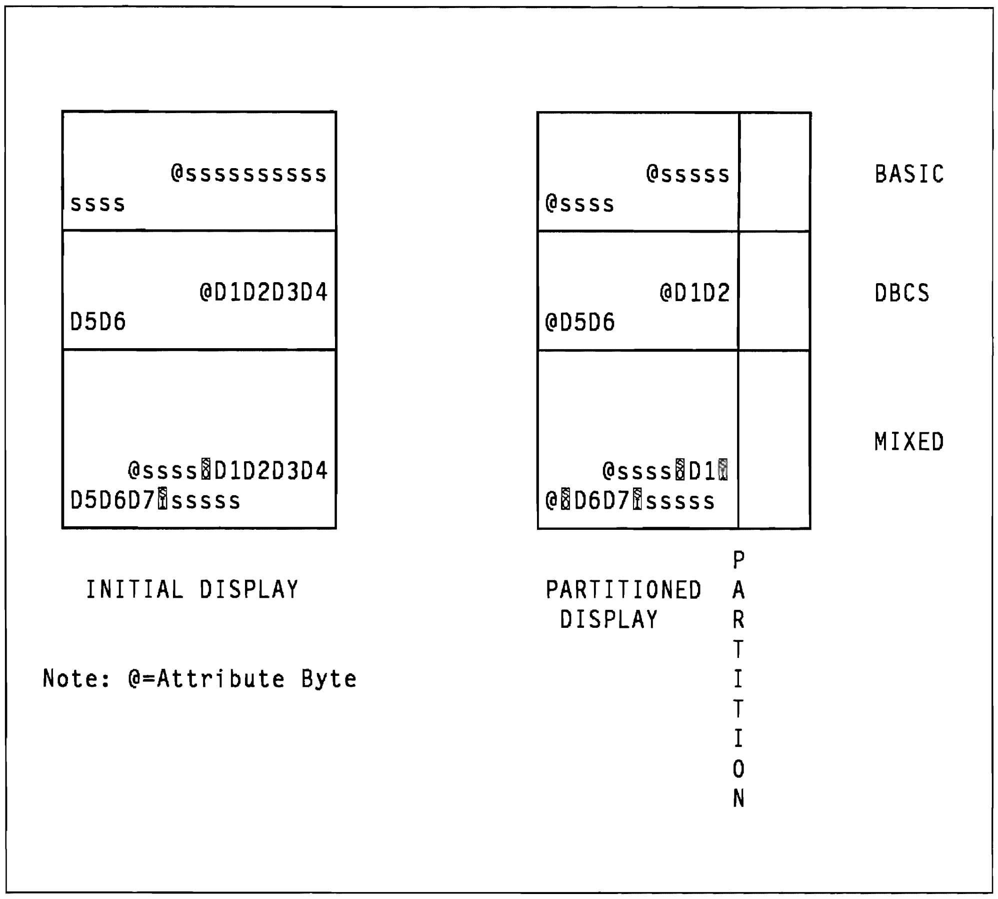


Figure 35. Example of Field Modification on Partitioned Display

- Compensation Rules for Displaying

  Figure 36 on page 95 shows the compensation rules (both Left Boundary and Right Boundary compensation rules) required for displaying a Mixed string that is overridden by other string. In this case, dummy attributes may be required to assure the validity of the Mixed string on a display, and to assure the integrity of the Mixed string in the storage. Dummy attributes are used to inhibit the user from inputting any character that causes a Mixed string in the storage to be invalid.

Left Boundary Compensation

* = Dummy Attribute

CASE	LEFT BOUNDARY		COMPENSATED RESULT	RULE REPLACE BY
1	D1	␏ss	*ss	Attribute
2	D	1␏ss	**ss	2 Attributes
3		D1␏ss	***ss	3 Attributes
4		D1D2	*␎D2	Attribute & SO
5	D	1D2	␎D2	SO
6		␎D1D2	␎D1D2	No adjustment

Right Boundary Compensation

CASE	RIGHT BOUNDARY		COMPENSATED RESULT	RULE REPLACE BY
1	D1␏	ss	D1␏	No Adjustment
2	D0D1	␏ss	D0␏*	SI & Attribute
3	D0D	1␏ss	D0␏	SI
4	s␎D	1D2	s**	2 Attributes
5	sss␎D1	D2	sss***	3 Attributes
6	sss␎	D1D2	sss*	1 Attribute

Figure 36. Compensation Rules for Scrolling : Displaying

- Compensation Rules for Integrity

  If the data is displayed in an output only field, no additional special handling is required.

  If it is displayed in an input field, additional special handling is required to maintain data integrity. To reflect changes made on the screen by the user, and then stored, two types of compensation are necessary depending on the user's updating of temporary SO/SIs. Figure 37 on page 96 and Figure 38 on page 96 show the special handling that may need to be done for Mixed data.

– Temporary SO or SI is not changed[51]

When storage is updated without any change of the temporary SO or SI used for display compensation, the temporary SO or SI is replaced by the original data.

```
                                      LEFT BOUNDARY
                                          |
                                          v
  ORIGINAL STORAGE                  s▒D1D2D3D4D5...
                                          |
  INITIAL SCREEN VIEW                     ▒D3D4D5...
                                          |
  USER UPDATE                             ▒D3DiD5...
                                          |
  INTERMEDIATE STORAGE              s▒D1D2D3DiD5...
                                          |
  UPDATED STORAGE                   s▒D1D2D3DiD5...

   Rule : Temporary SO, SI is replaced by original data.
```

Figure 37. Example of Temporary SO/SI (1)

– Temporary SO or SI is changed

When storage is updated after the user changes the temporary SO or SI for display compensation, the original data must be adjusted to maintain data integrity.

```
                               LEFT BOUNDARY     LEFT BOUNDARY
                                    |                 |
                                    v                 v
  ORIGINAL STORAGE             s▒D1D2D3D4...     s▒D1D2D3D4...
                                    |                 |
  INITIAL SCREEN VIEW               ▒D3D4...         *▒D3D4...
                                    |                 |
  USER UPDATE                       ss▒D4...         *ss▒D4...
                                    |                 |
  INTERMEDIATE STORAGE         s▒D1Dss▒D4...     s▒D1Dss▒D4...
                                    |                 |
  UPDATED STORAGE              s▒D1▒ss▒D4...     s▒D1▒ss▒D4...
```

Figure 38. Example of Temporary SO/SI (2)

Note: In the above example, data is destroyed that the user cannot see. Some warning such as highlighting might be issued. When contiguous SO/SI is generated after the adjustment, it should not be eliminated but replaced by blanks to keep the position as is.

[51] SO or SI added to the device data stream for boundary adjustment is called a temporary SO, SI.

Appendix C. Common DBCS Processing Functions

Applications may require more functions to manipulate DBCS data beyond those supported by high level languages or utilities. Then it is often useful to provide the additional functions for DBCS data as subroutines that various applications can call. In this appendix, the following 13 functions which may meet the requirements of applications are introduced. It is recommended that you select the necessary functions for your applications from them and build a set of your own subroutines.

- Pre-processing
  - VALIDATION
  - ADD SO/SI
  - DELETE SO/SI
- Processing
  - ASSIGNMENT (Move)
  - SUBSTRING
  - CONCATENATION
  - SPLIT
  - DECOMPOSE
  - SEARCH
  - CHANGE
  - CONVERT
  - ANALYZE
- Post-processing
  - ADJUST

Note that some functions are provided as base functions of some high level language or utility, so all of the subroutines described here do not necessarily have to be implemented.

Note: The sample syntaxes referred to here are ones of typical syntaxes that sufficiently meets the requirements of general applications. The implementation, whether or not you adopt each syntax, depends on your application requirements.

Parameters commonly used in the sample syntaxes are:

INSTR	Specifies the input string name.
INLEN	Specifies the input string length.
INTYP	Specifies the input string type, either MIXED or DBCS.

OUTSTR	Specifies the output string name.
OUTLEN	Specifies the output string length.
OBYTE	Specifies the buffer name where the byte length of output string will return.
OCHAR	Specifies the buffer name where the logical character length of output string will return.
CNTMODE	Specifies the counting mode, either byte counting or logical character counting.
REPCHAR	Specifies the character that replaces the invalid character. When the invalid character is a DBCS character, the replace character (X′xx′) is converted to 42nd ward DBCS character (X′42xx′).
JUST	Specifies the justification rule.
PAD	Specifies the character that is used for padding.
BYTEPOS	Specifies the buffer name where the position counted by physical bytes will return.
CHARPOS	Specifies the buffer name where the position counted by logical characters will return.
RCODE	Specifies the buffer name where the return code will be set.
REASON	Specifies the buffer name where the reason code will be set.

VALIDATION

This subroutine verifies that an input string identified as DBCS is a valid DBCS data string or an input string identified as Mixed is a valid Mixed data string. Both of the two types of strings should be validated under the criteria described in "Validity Check" on page 21. In case that the string is invalid, this subroutine compensates it and returns to the caller a return/reason code.

```
Sample syntax : DBVALID INSTR(input_str) [INLEN(input_len)]
                        [INTYP('DBCS'|'MIXED')]
                        [REPCHAR(X'xx'|X'40')]
                        [OUTSTR(output_str)] [OUTLEN(output_len)]
                        [RCODE(return_code)] [REASON(reason_code)]
```

Note: Performance Consideration
Code check may not be mandatory if only the performance is the key consideration.

ADD SO/SI

This subroutine converts a given valid DBCS string to a Mixed string (bracketed DBCS string) by adding an SO character to the beginning and SI character to the end of the DBCS string. If an input DBCS string is invalid, the subroutine will return to the caller a return/reason code.

```
Sample syntax : DBADDSOSI INSTR(input_str) [INLEN(input_len)]
                          [OUTSTR(output_str)] [OUTLEN(output_len)]
                          [RCODE(return_code)] [REASON(reason_code)]
```

DELETE SO/SI

This subroutine converts a given valid bracketed a DBCS string to DBCS string by removing an SO/SI pair from it. If the bracketed DBCS string is invalid or any other construct is input, the subroutine will return to the caller a return/reason code.

```
Sample syntax : DBDELSOSI INSTR(input_str) [INLEN(input_len)]
                          [OUTSTR(output_str)] [OUTLEN(output_len)]
                          [RCODE(return_code)] [REASON(reason_code)]
```

ASSIGNMENT (MOVE)

This subroutine moves one string to another string. The sending and receiving strings can be of the same or different lengths. This subroutine handles the following types of moves:

- DBCS to DBCS
- Mixed to Mixed

If the output string is shorter than the input data, truncation will be performed on the output string to maintain data integrity. If the output string is larger than the input data, it will be padded to the right or left, depending on the parameter for justification, with the padding character (the default padding character is a blank). Refer to 3.0, "Processing DBCS" on page 13 for the truncation, padding rule.

```
Sample syntax : DBMOVE INSTR(input_str) [INLEN(input_len)]
                       [INTYP('DBCS'|'MIXED')]
                       [OUTSTR(output_str)] [OUTLEN(output_len)]
                       [JUST('ASIS'|'LEFT'|'RIGHT')]
                       [PAD(pad_char)]
                       [RCODE(return_code)] [REASON(reason_code)]
```

SUBSTRING

This subroutine extracts data from a given string. It operates on either DBCS or Mixed strings. This subroutine adjusts the data at the start and end of the substring to maintain character boundaries. The substring rule of the subroutine is based on the truncation rule described in "Truncation" on page 17. It is specifiable that the unit of the specified start position and length is "physical byte length" or by "logical character length" .[52]

[52] Physical byte counting counts in terms of the number of bytes, including SO/SI. Logical character counting counts in terms of the number of SBCS or DBCS characters, skipping SO/SI.

```
Sample syntax : DBSUBSTR INSTR(input_str) [INLEN(input_len)]
                         [INTYP('DBCS'|'MIXED')]
                         [OUTSTR(output_str)] [OUTLEN(output_len)]
                         STARTPOS(start_pos)
                         LENGTH(length)
                         [OBYTE(output_byte)] [OCHAR(output_char)]
                         [CNTMODE('BYTE'|'CHAR')]
                         [RCODE(return_code)] [REASON(reason_code)]
```

CONCATENATION

This subroutine concatenates two substrings into a single string and returns the resultant concatenated string in the output string. Basically, the concatenation is performed between two input strings having the same data type (DBCS or Mixed). When you want to concatenate two input strings that have different data types, ADD SO/SI, DELETE SO/SI, or CONVERT subroutine is required to adjust the input string in advance.

If the data type of the resultant string is Mixed, the user may optionally supply the appropriate processing option parameter to adjust it as described in "Adjacent SO/SI Pair" on page 19.

```
Sample syntax : DBCONCAT INSTR1(input_str1) [INLEN1(input_len1)]
                         INSTR2(input_str2) [INLEN2(input_len2)]
                         [INTYP('DBCS'|'MIXED')]
                         [OUTSTR(output_str)] [OUTLEN(output_len)]
                         [OBYTE(output_byte)] [OCHAR(output_char)]
                         [OPTION('ADJREPL'|'ADJREM')]
                         [RCODE(return_code)] [REASON(reason_code)]
```

SPLIT

This subroutine divides a DBCS or Mixed string into two valid substrings. The split starting position can be specified in either logical characters or physical bytes.

This subroutine adjusts the data at the start of the split to maintain character boundaries when necessary. Also, if the starting position is located in a DBCS portion of a Mixed string, the resulting output string will have the SI and SO characters added, as necessary, to insure the integrity of the output strings.

```
Sample syntax : DBSPLIT INSTR(input_str) [INLEN(input_len)]
                        [INTYP('DBCS'|'MIXED')]
                        [OUTSTR1(output_str1)] [OUTLEN1(output_len1)]
                        OUTSTR2(output_str2) [OUTLEN2(output_len2)]
                        STARTPOS(start_pos)
                        [OBYTE1(output_byte1)] [OCHAR1(output_char1)]
                        [OBYTE2(output_byte2)] [OCHAR2(output_char2)]
                        [CNTMODE('BYTE'|'CHAR')]
                        [RCODE(return_code)] [REASON(reason_code)]
```

DECOMPOSE

This subroutine is used to identify the data type, start position and length of an SBCS or DBCS portion of a valid Mixed string.

"Portion number" shows a relative position of the substring.

```
                         a␎D1D2␏b␎D3␏
                         | |    | |
  portion number  :      1 2    3 4
```

The construct of a Mixed string can be determined by calling this subroutine iteratively; the first time to locate the decomposed portion specified by the portion number and return its type (SBCS, DBCS or NULL[53]), start position and length, and subsequently to locate the next decomposed portion and return its type, start position and length. If the specified portion number is greater than the number of portions in the Mixed string, the subroutine will return to the caller a return/reason code.

The starting position and length of the decomposed portion can be returned in physical bytes and/or logical characters.

```
Sample syntax : DBDECOMP INSTR(input_str) [INLEN(input_len)]
                         PTYPE(portion_type) [PNUM(portion_num)]
                         BYTEPOS(byte_pos) [OBYTE(output_byte)]
                         [CHARPOS(char_pos)] [OCHAR(output_char)]
                         [RCODE(return_code)] [REASON(reason_code)]
```

SEARCH

This subroutine finds where a user specified search string occurs within a given string and returns the position in which the string occurs. Both input strings may be either DBCS or Mixed strings.

If a match occurs, the starting position can be returned in physical bytes and/or logical characters. Then leading and trailing blanks count as data. If a match does not occur, a return/reason code will be set.

```
Sample syntax : DBSEARCH INSTR1(input_str1) [INLEN1(input_len1)]
                         [INTYP1('DBCS'|'MIXED')]
                         INSTR2(input_str2) [INLEN2(input_len2)]
                         [INTYP2('DBCS'|'MIXED')]
                         [BYTEPOS(byte_pos)] [CHARPOS(char_pos)]
                         [RCODE(return_code)] [REASON(reason_code)]
```

[53] "NULL" indicates that the portion consists of a contiguous SO/SI.

CHANGE

This subroutine replaces the first or all occurrences of a subset of DBCS or Mixed characters (string1) within a given string (string3) with another set of characters (string2). If string1 is not found in string3, a return/reason code will be set.

The following are the basic rules for determining the data type of the output string.

string1	string2	string3	→	output string
DBCS	DBCS	DBCS		DBCS
DBCS	DBCS	MIXED		MIXED
DBCS	MIXED	DBCS		MIXED
DBCS	MIXED	MIXED		MIXED
MIXED	DBCS	DBCS		error
MIXED	DBCS	MIXED		MIXED
MIXED	MIXED	DBCS		error
MIXED	MIXED	MIXED		MIXED

```
Sample syntax : DBCHANGE INSTR1(input_str1) [INLEN1(input_len1)]
                         [INTYP1('DBCS'|'MIXED')]
                         INSTR2(input_str2) [INLEN2(input_len2)]
                         [INTYP2('DBCS'|'MIXED')]
                         INSTR3(input_str3) [INLEN3(input_len3)]
                         [INTYP3('DBCS'|'MIXED')]
                         OUTSTR(input_str) [OUTLEN(input_len)]
                         [OBYTE(output_byte)] [OCHAR(output_char)]
                         [OPTION('FIRST'|'ALL')
                         [RCODE(return_code)] [REASON(reason_code)]
```

CONVERT

This subroutine converts the SBCS portions of a Mixed string to DBCS portions by inserting a X'42' before each SBCS byte (SBCS space is converted to DBCS space), or converts the 42nd ward DBCS portions of a DBCS or Mixed string to SBCS portions by removing the first byte (DBCS space is converted to SBCS space). The return/reason codes show whether the resultant string consists of only SBCS characters in case of the conversion of DBCS string to Mixed.

```
Sample syntax : DBCONVERT INSTR(input_str) [INLEN(input_len)]
                          [INTYP('DBCS'|'MIXED')]
                          OSTR(input_str) [OLEN(input_len)]
                          [OBYTE(output_byte)] [OCHAR(output_char)]
                          [RCODE(return_code)] [REASON(reason_code)]
```

ANALYZE

This subroutine indexes to the next physical byte or logical character in a Mixed string and maintains a type indicator (SO, SI, SBCS, DBCS1, or DBCS2[54]) of

[54] DBCS1/DBCS2 indicates the first or the second byte of a DBCS character respectively

the byte at the current index position. This subroutine is invoked iteratively; the first time to locate the specified index and return the type of the byte at that index, and subsequently to increment the index to the next position and return the type of the byte at that index.

```
Sample syntax : DBANALYZE INSTR(input_str) [INLEN(input_len)]
                          BYTEPOS(byte_pos) CHARPOS(char_pos)
                          BTYPE(byte_type)
                          [INDXMODE('BYTE'|'CHAR')]
                          [OPTION('GIVEN'|'NEXT')]
                          [RCODE(return_code)] [REASON(reason_code)]
```

ADJUST

This subroutine removes unnecessary adjacent SI/SO and SO/SI pairs from a valid Mixed string.

There are two types of removal; removal of all pairs found and replace of all pairs found with X'4040'.

This subroutine optionally returns to the user a count of the number SI/SO and SO/SI pairs removed or replaced.

```
Sample syntax : DBADJUST INSTR(input_str) [INLEN(input_len)]
                         [OSTR(input_str) [OLEN(input_len)]]
                         [OBYTE(output_byte)] [OCHAR(output_char)]
                         [COUNT(count)]
                         [OPTION('ADJREPL'|'ADJREM')]
                         [RCODE(return_code)] [REASON(reason_code)]
```

Appendix D. DBCS Testing

Most of the differences between SBCS and DBCS handling are absorbed by the many IBM-supplied DBCS-enabled products such as programming languages, data base managers, and presentation service products as seen in the previous chapters. This makes application program testing much easier in terms of DBCS, that is, attention does not have to be paid to the detailed process of DBCS data. Also, knowledge about supported national languages is not necessary as far as functional testing is concerned.

The following are some points when testing applications with DBCS capabilities.

D.1 Systems Environment

DBCS testing cannot be done without a proper hardware and software installation. In addition to the host environment, the 3270 PC or 3270 PC/G program must be properly customized by the workstation user. This customization specifies various terminal characteristics such as the number of display sessions, response time monitoring, or terminal printer characteristics. For Japanese 3270 PC(/G), the SBCS basic character mode (Japanese Katakana or U.S. English) is specified in this customization.

D.2 Language Knowledge - Functional Testing and Translation Testing

DBCS functional testing does not assume any particular national character set to use. It can utilize any DBCS character set supported by the workstation. This means testing does not have to be done over all the DBCS character sets, for example, DBCS testing will be complete by using just one DBCS character set. The 3270 PC(/G) on the PS/55 supports one SBCS and one DBCS at a time. Recommendation is to use DBCS (English) alphabets that reside in every DBCS character set's 42nd ward. This will simplify the testing efforts. DBCS alphabets can be easily entered using the alphabetic keys and do not need the national language-dependent input methods. Such DBCS alphabets will be enough for almost all the test cases, however, if testing needs a DBCS national character, one suggestion is to use the hexadecimal input. Language-dependent input methods can also be used.

DBCSs differ in size. Although the general DBCS code scheme is defined as in Figure 2 on page 4, up to which code point DBCS characters are defined depends on each language.

Contrary to functional testing, translation testing does need language knowledge. It consists of translation verification, MRI (Machine Readable Information) integration, verification of DBCS substitution, and panel direction through a session.

D.3 Some Technical Considerations

The following are some detailed DBCS handling considerations which an application program may need to test. Each application design should carefully examine the DBCS support by its prerequisite products. Testing on the following items should be done depending on what the prerequisite supports.

- Syntactical code point handling

  Since a DBCS character allows every code between X'41' and X'FE' in its first or second byte (DBCS blank is the exception X'4040'), syntactical SBCS character codes can be in a DBCS character. For example, SBCS '/' (slash) is often used as a delimiter. Its hexadecimal value X'61' could be in a DBCS character like X'xx61'. When a parser parses a string that has such a DBCS character, it should not be recognized as a delimiter. Similar attention should be paid to other codes that are often used as syntactical characters.

  ```
  Examples;

     CHANGE/␎D1D2␏/ab␎D3D4␏/       D2=X'xx61'   /(slash)=X'61'
     'abc␎D1D2␏'                   D1=X'7Dxx'   '(quote)=X'7D'
  ```

- Mixed strings handling

  There are some IBM-supplied products in which operations on a Mixed string are done on a byte basis. Since such byte-oriented operations may produce an invalid DBCS portion, the application program itself often supports the function to compensate such invalid strings. Testing should cover such operations. These operations are;

  - Assignment (Move)
  - Substring
  - Split

  See 3.2.2, "Operation for Mixed String" on page 16.

- Upper-casing and lower-casing

  DBCS languages do not have a upper/lowercase concept. Upper-casing and lower-casing operations should skip such national characters. Some products may allow upper/lower-casing for the 42nd ward (English alphabet) characters.

- SO/SI on display and print-out

  On a PS/55 display, when using the 3270PC(/G) program, shift-codes are displayed as blanks by default, however, the codes can also be visualized through the workstation control mode of 3270PC(/G) program running on the OS/2-J environment.[55] Testing should use it. Otherwise, for example, the following two mixed strings are seen to be identical;

  ```
  ␎D1D2D3ƀƀƀƀD4D5␏
  ␎D1D2D3␏ƀƀ␎D4D5␏
  ```

  Similarly, printed fonts for the shift-codes can be specified through some printing programs such as the Kanji Utility.

[55] For 3270PC(/G) program runnings under the DOS environment, the codes can be visualized by pressing the PF1 key while holding Control (Ctrl) key.

Appendix E. PS/55

E.1 Overview

The IBM PS/55, which is comparable to the IBM PS/2 in DBCS countries, is a multi-purpose workstation that provides DBCS input/output capability as:

- 3270 DBCS Terminal
- 5250 DBCS Terminal for AS/400, S/36 and S/38
- Personal Computer / Word Processor

The IBM PS/55 consists of Models 5550, 5530, 5535, 5540, 5560, 5570, and the related software. The system provides a DBCS input conversion sub-system that is used for DBCS character input from a standard keyboard. The system also provides a DBCS output capability for a display and a printer.

The IBM PS/55 has been available in some Asian countries, including Japan, where the languages require DBCS. Hardware consists of a system unit, a display, a keyboard, and a printer. The keyboard and font card in the system unit are unique to each country's version of the IBM PS/55 family and remaining components are common to all country versions of the IBM PS/55. The principal software are 3270PC or 3270PC/G for MFI, DOS or OS/2™-J (Japanese version of OS/2)[56] for personal computers, and word processors. The software is unique to each country's version of the IBM PS/55.

As a 3270 DBCS terminal, it can communicate to the host system via SNA/SDLC, BSC, IBM token-ring, or X.25. The file transfer between the host system and PC is available.

E.2 PS/55 Display

The IBM PS/55 display is high-resolution display (1024 × 768 dots), which provides the all-points-addressable function in graphic mode, and DBCS output capability. As a 3270 DBCS terminal, this display can handle the 3270 extended data stream and provides the following fields.

- SBCS (EBCDIC) field
- DBCS field
- MIXED (SO/SI creation enabled) field
- Numeric field [57]

[56] IBM OS/2 is a trademarks of IBM Corporation.

[57] The customization of 3270 PC or PC/G is required to define numeric fields.

E.3 PS/55 Printer

Various IBM PS/55 printers that have DBCS capability can be attached via the external interface. Users can choose hardware in accordance with their requirements such as a processing speed, a color/monochrome function, a printing quality, based on application requirements.

E.4 3270 PC

The Japanese 3270 PC provides for 3270 operation on the IBM PS/55 for either clustered or single station communication[58] to appropriately programmed IBM System/370, 93xx, 43xx, 30xx, and 8100 systems. The highlights are summarized below:

- 3270 display and printer session with DBCS (LU-1, LU-2, LU-3 and non-SNA).
- Field outlining (Grid line) using 3270 extended data stream.
- Extended color (7 colors) and extended highlighting on either a field or character basis.
- For a single-byte character set, US English or a Japanese Katakana character set is selectable at customization time.
- Printer sharing among the clustered IBM PS/55 family of workstation attached to a 3274/3174 Control Unit for local copy operation.
- Switching of the display screen ownership by the HOT key between a MFI session and a PC application session in a flip-flop manner.
- File transfer between the host system and the PC.
- Multiple logical terminal support.

Also Korean, Traditional Chinese, and Simplified Chinese versions are available.

[58] A proper communication adapter card is required for the 3270PC according to the type of host-PC communication.

E.5 3270 PC/G

The Japanese 3270 PC/G offers users the same functions as the Japanese 3270 PC and is extended to manipulate and interact with graphics.

The host interactive graphics function on the IBM System/370, 43xx, and 30xx processor is supported by GDDM Release 4 or later. The highlights are summarized below:

- APL support

  Provides the output and input of APL specific characters for both VS APL and APL 2.

- Plotter support

  Supports for IBM 7371 and 7372 Plotters which are controlled through GDDM from the host.

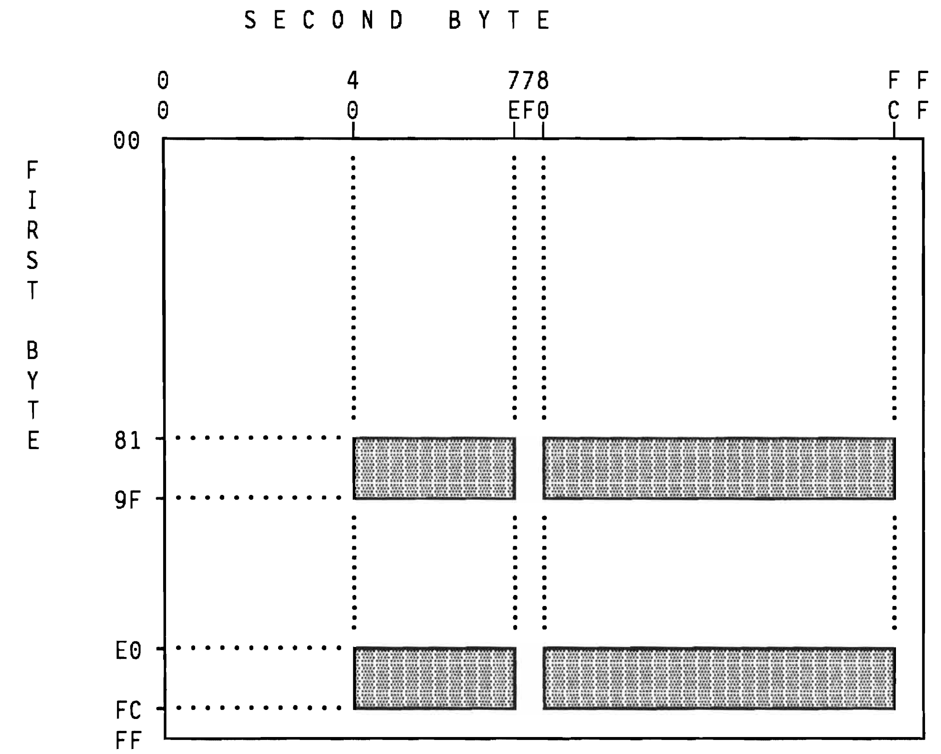
E.6 DBCS Input Method

In order to input DBCS characters, it is always necessary to enter multiple key strokes. After the key strokes are entered, a choice from a set of eligible candidates may have to be made by the operator. The DBCS input method depends on the uniqueness of each DBCS language. For instance, Kana to Kanji conversion in Japanese, Hanguel Components to Hanguel in Korean, Radical Components to Chinese character in Simplified/Traditional Chinese. Host applications are independent of the input method for entering DBCS characters. That is, any host application does not need to pay attention to the difference of the input methods.

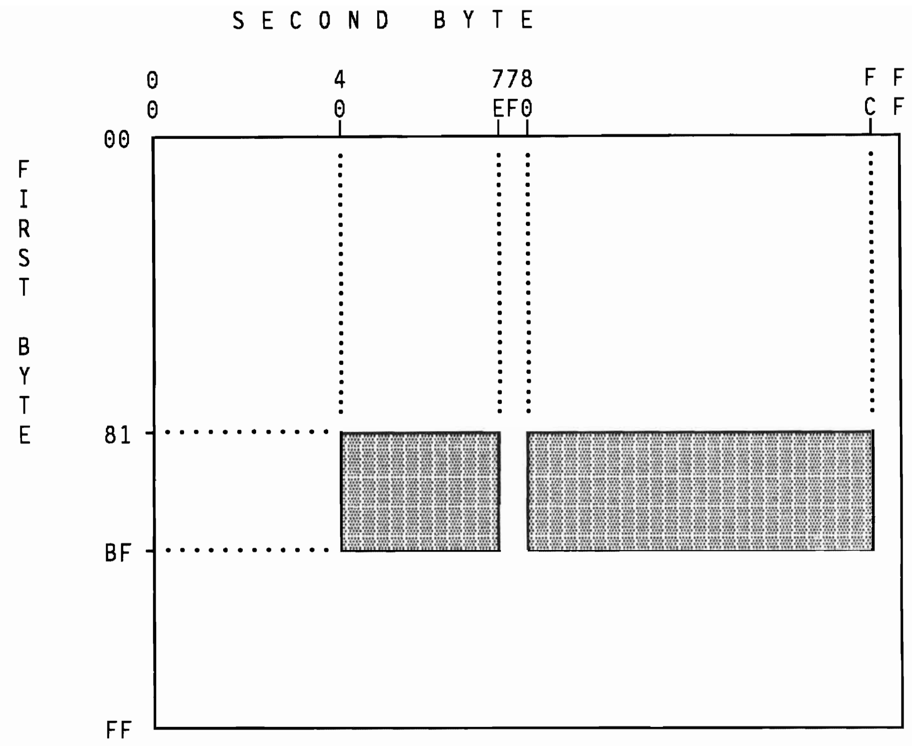
Appendix F. DBCS-PC

The DBCS-PC code does not use control characters such as SO/SI to identify DBCS characters. Instead, it uses a different code structure, in which the values of the first byte of the DBCS characters are not used as values for the SBCS character. Thus, in the DBCS-PC code set, each character is either a SBCS or a DBCS character and is self-defining.

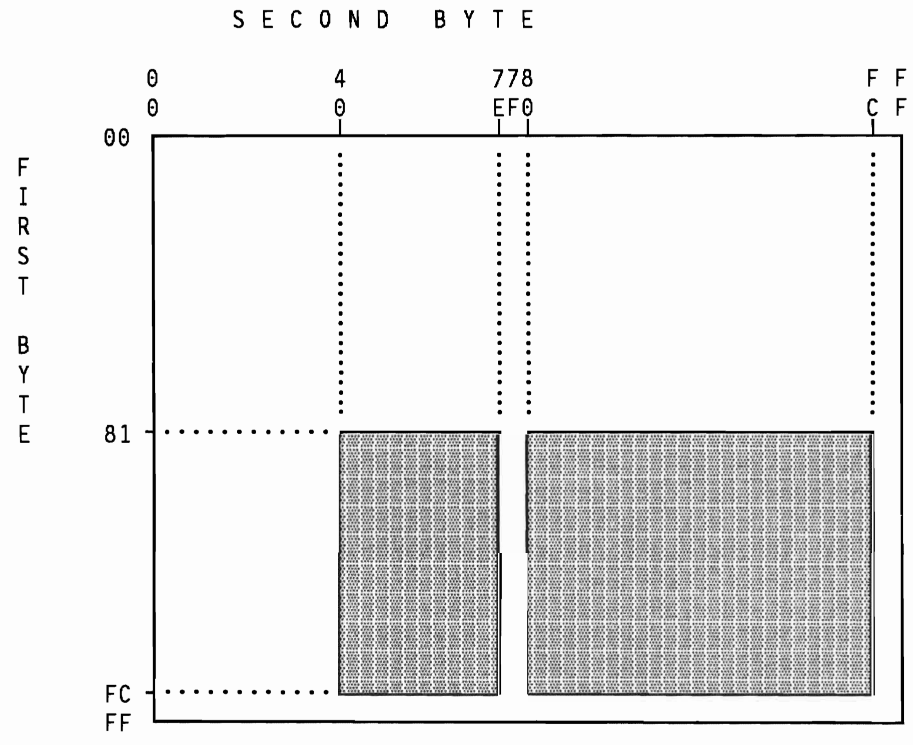
The following charts show the DBCS-PC code structure for each DBCS country.



▨ : DBCS-PC Code Area for Japanese



▨ : DBCS-PC Code Area for Korean



▨ : DBCS-PC Code Area for Simplified/Traditional Chinese

Appendix G. Product List

This appendix contains the list of IBM-supplied products referred to in this manual. The version/release number of each product shows the first version/release that supports DBCS functions described in this manual.

Presentation Service Product

Product	OS	Ver.	Rel.	Mod.	Remarks
ISPF/DM	MVS VM	2 2	1 2	1 0	No DBCS support in VSE version
GDDM	MVS/VM/VSE	1 2	4 1	0 0	 Mixed field without SO/SI
CICS/BMS	MVS/VSE	1	7	0	V1R6 with DBCS feature
IMS/MFS	MVS	2	2	0	V1R3/V2R1 with DBCS feature

Application Generator/Map Definition

Product	OS	Ver.	Rel.	Mod.	Remarks
CSP/AD/AE	MVS/VM/VSE	3	2	0	
SDF II	MVS/VM	1	1	0	

Printer Service Product

Product	OS	Ver.	Rel.	Mod.	Remarks
PSF	MVS/VSE VM	1 1	1 1	1 0	
RSCS	VM	2	2	0	
Kanji Utility	MVS/VSE	2	1	0	
KDP	MVS	2	2	0	

Language

Product	OS	Ver.	Rel.	Mod.	Remarks
VS COBOL II	MVS/VM	1	2	0	
OS PL/I	MVS/VM	2	1	0	
VS FORTRAN	MVS/VM	2	3	0	
Assembler H	MVS/VM	2	1	0	

Data Base Manager

Product	OS	Ver.	Rel.	Mod.	Remarks
DB2	MVS	1	2	0	
SQL/DS	VM/VSE	1	3	0	
IMS/DB, DL/I	MVS	2	2	0	

Glossary

A

Advanced Function Printing (AFP). The ability of the program to use the APA (all-points-addressable) concept to print text and illustrations on printers.

B

bracketed DBCS string. Mixed string which consists of DBCS characters (no SBCS character) and a pair of SO/SI characters.

C

CGCSGID. Coded Graphic Character Set Global Identifier.

character data type. The characteristics of a data items that are used when a processing program handles a string as a character string.

CICS/BMS. Customer Information Control System/Basic Mapping Support.

COBOL. Common Business Oriented Language.

CSP. Cross System Product.

D

DBCS. See Double-Byte Character Set.

DBCS country. Asian countries where their languages are represented using DBCS characters.

DBCS data. General term which stands for both DBCS string and Mixed string.

DBCS data type. The characteristics of a data item, which are used when a program handles a string as a DBCS string.

DBCS field. A field on a DBCS workstation which displays and accepts only DBCS strings.

DBCS-HOST. DBCS defined in the S/370 and S/3x environment, whose code is based on EBCDIC.

DBCS-PC. DBCS defined in the PC environment, whose code is based on ASCII.

DBCS string. Character string that consists of DBCS characters only.

DB2. Database 2. Relational data base manager running under the MVS environment.

DL/I. Data Language/I

Double-Byte Character Set (DBCS). A ideographic character set that uses a double-byte coding scheme to represent some Asian languages that have several thousand symbols.

E

extended attribute. Additional attribute provided by the 3270 extended data stream for example SO/SI creation, field outlining, or extended highlightning.

EBCDIC field. See SBCS field.

F

field outlining. Ruled line to show data in tabular form or identify where input fields are located.

FORTRAN. Formula Translation (programming language).

G

GCSGID. Graphic Character Set Global Identifier, which is an identifier assigned to distinguish each graphic character set.

GDDM. Graphic Data Display Manager.

Graphic data type. Same meaning as DBCS data type. DBCS data type is implemented as Graphic data type in some products.

I

IBM PS/55 family. DBCS capable workstation that is comparable to IBM PS/2.

IMS/MFS. Information Management System/Message Format Service.

ISPF/DM. Interactive System Productivity Facility/Dialog Manager.

J

Japanese-Katakana. Japanese unique character set that contains uppercase alphabet and SBCS Katakana instead of the lowercase alphabet. SBCS Katakana has been used traditionally in DP applications. Katakana also is defined in DBCS as one component of the Japanese written language with Kanji.

K

Kanji. Japanese ideographic character set. The term "KANJI" is sometimes used synonymously with "DBCS".

Kanji Data Set Print program (KDP). The printer support software that provides the function for printing DBCS characters and drawing field outlining on both terminal printers and system printers. This software can be used commonly among Asian countries.

Kanji Utility. A set of utility programs provided for DBCS processing. The Kanji Utility consists of Kanji Print Utility, COBOL Preprocessor, Two-Byte Code Conversion, and CICS/BMS Preprocessor. These utilities can be used commonly among Asian countries.

Kanji Print Utility. One of the components of Kanji Utility. This program accepts DBCS data sets, edits them according to the utility control statement, and prints them on the IBM 3800 printing subsystem.

Katakana. Japanese phonetic character set.

Katakana terminal. Displays that are installed with the Japanese Katakana feature. The lowercase alphabetic is presented as Katakana characters on this display.

KDP. See Kanji Data Set Print program.

M

machine readable information (MRI). All textual information contained in a program, which may appear on display panels or printers.

Mixed field. A field on a DBCS workstation which is enabled to input Mixed string by operators.

Mixed string. Synonym of "SBCS/DBCS mixed string".

multilingual support. Multiple language support for MRI (Machine Readable Information) issued from a product. This is usually the support of bilingual MRI for SBCS terminals and DBCS terminals.

N

national language support (NLS). To translate MRI in the native language and integrate translated MRI.

P

PL/I. Programming Language/I.

PSF. Print Services Facility. One of AFP support program. Prerequisite for DBCS data printing.

PS/55 family. see IBM PS/55 family.

R

RSCS. Remote Spooling Communications Subsystem.

S

SBCS data type. Same meaning as "character data type".

SBCS/DBCS mixed string. Character string that consists of both SBCS and DBCS characters enclosed with SO/SI.

SBCS field (EBCDIC field). A field on a DBCS workstation which is basically equivalent to the current 3270 field.

SBCS string. Character string that consists of SBCS characters only

SDF II. Screen Definition Facility II.

Shift-out, Shift-in (SO/SI). Special delimiters to identify SBCS and DBCS characters. Shift-out (X'0E') indicates the beginning of DBCS and Shift-in (X'0F') indicates the end of DBCS.

Simplified Chinese. The language used in People's Republic of China (PRC).

Single-byte character set(SBCS). Graphic character set that uses single byte coding scheme.

SO/SI. See Shift-out, Shift-in.

SQL/DS. Structured Query Language/Data System. Relational data base manager running under VM and VSE environment.

T

Traditional Chinese. The language used in Republic of China (ROC).

V

VTAM. Virtual Telecommunication Access Method.

Numeric

3270 extended data stream. One of device data streams that provides the capability to define extended attributes of field or character such as SO/SI creation or field outlining.

42nd word DBCS. Double-byte alphanumeric or special symbols, whose first byte is X'42' and second byte is the same hex value of a corresponding EBCDIC code.

Index

IBM DBCS Design Guide
- System/370 Software
GG18-9095-0

READER's COMMENT FORM

This manual is a part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Possible topics for comments are: Clarity, Accuracy, Completeness, Organization, Retrieval, and Legibility.

If you wish a reply, give your name and mailing address.

Name: ______________________________

Address: ______________________________

Occupation: ______________________________

Comments: ______________________________

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

GG18-9095-0

Reader's Comment Form

Fold and Tape **Please Do Not Staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO.40 ARMONK, N.Y. U.S.A.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
DBCS Competency Center, Dept A7J, Bldg R19A
PO Box 28331
Atlanta, GA 30358-0331

Fold and Tape **Please Do Not Staple** Fold and Tape

IBM®

IBM DBCS Design Guide
- System/370 Software
GG18-9095-0

READER's COMMENT FORM

This manual is a part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Possible topics for comments are: Clarity, Accuracy, Completeness, Organization, Retrieval, and Legibility.

If you wish a reply, give your name and mailing address.

Name: ______________________________

Address: ______________________________

Occupation: ______________________________

Comments: ______________________________

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

GG18-9095-0

Reader's Comment Form

Fold and Tape **Please Do Not Staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO.40 ARMONK, N.Y. U.S.A.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
DBCS Competency Center, Dept A7J, Bldg R19A
PO Box 28331
Atlanta, GA 30358-0331

Fold and Tape **Please Do Not Staple** Fold and Tape

IBM®

IBM DBCS Design Guide
- System/370 Software
GG18-9095-0

READER's COMMENT FORM

This manual is a part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Possible topics for comments are: Clarity, Accuracy, Completeness, Organization, Retrieval, and Legibility.

If you wish a reply, give your name and mailing address.

Name: ______________________________

Address: ______________________________

Occupation: ______________________________

Comments: ______________________________

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

GG18-9095-0

Reader's Comment Form

Fold and Tape **Please Do Not Staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO.40 ARMONK, N.Y. U.S.A.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
DBCS Competency Center, Dept A7J, Bldg R19A
PO Box 28331
Atlanta, GA 30358-0331

Fold and Tape **Please Do Not Staple** Fold and Tape

IBM®

GG18-9095-00

Printed in U.S.A.