Volume 5, Number 6 May 1987

IBM Personal Computer Multilingual Code Page 850
Application Development Considerations

IBM Personal System/2™ Seminar Proceedings

The Publication for Independent Developers
of Products
for IBM Personal System/2

Published by International Business Machines Corporation
Entry Systems Division

Printed in the
United States
of America

Contents

Foreword

IBM Personal System/2™ Seminars and Proceedings provide information about new product announcements and enhancements to existing products, and are intended to assist independent developers in their hardware and software development efforts.

Over the past several years, the success of the IBM Personal Computer family was due in part to the efforts of independent developers, whose hardware and software products have become widely used. For its part, IBM helped these vendors by holding relevant technical seminars and publishing the proceedings of those seminars. The result was a mutually beneficial partnership and transfer of technical knowledge.

With the advent of the Personal System/2 family, IBM's seminar program will continue. Through these seminars and the corresponding Proceedings, IBM will address the independent developers' need for technical information about the latest IBM products. In these and future Proceedings, you will find technical information about subjects such as:

- IBM computer design and architecture
- IBM computer components and their interaction
- Memory capacities, speeds, transfer rates
- Input/output device capacities, speeds, access methods and rates
- Graphics and display technologies, programming considerations
- Printing technologies, programming considerations
- Operating system high level interfaces
- Development tools: capabilities, languages, program verification aids
- Compatibility considerations
- Communications: capabilities, offerings, statistics
- Enhancements to existing IBM hardware and software products
- Hints, tips and techniques to enhance your productivity

Through these seminars and proceedings, IBM intends to maintain its partnership with independent developers and assist them in successfully producing hardware and software products for the IBM Personal System/2 Family.

Personal System/2 is a trademark of the IBM Corporation.

Code Page 850

Introduction

IBM has announced a new code page and character set available for its IBM Personal Computers and IBM Personal System/2™. IBM DOS Version 3.30 and follow-on operating systems now have the ability to switch between the present and future code pages. This document is intended to clarify the reasons for introducing the IBM Personal Computer Multilingual Code Page 850 (abbreviated to MLP CP850 or just 850) and to provide advice and guidance to developers of IBM PC application software.

The document is divided into four major sections:

- The reasons for the introduction of code page 850; the requirement for code page switching and its benefits.
- Guidelines for writing new applications.
- Guidelines for changing existing applications.
- National language items.

The section "Writing New Applications" offers advice about how to obtain the maximum benefit from CP850. It also discusses how to minimize the difficulties of migrating to code page 850 once you have started coding your application using one of the current PC code pages (437, 860, 863, or 865).

An existing application modified to operate under code page 850 is almost certain to conflict with some of the guidelines for writing new applications. The section "Changing Existing Applications" is a checklist of questions to help determine the extent to which an existing application is affected, and provide advice about dealing with the problems that are identified.

The final section of this document, "National Language Items" on page 14, contains general advice and guidance on writing applications for the international market.

Throughout these proceedings a number of key principles are highlighted in boxes. For convenience, these also are listed at the end of the publication.

Background of Code Page 850 and Code Page Switching

The IBM Personal Computer has matured from a single-user system for personal productivity applications to a multipurpose, communicating workstation capable of executing a range of administrative, business and communications applications. This has led to a requirement for more effective communication and data exchange between IBM PCs and between IBM PCs and host mainframes. Common multilingual code pages fulfill this requirement.

The industry standard that satisfies this requirement is the 8-bit code page defined by ISO as Standard DIS 8859. This code page fully supports the alphabetic characters of all Latin-based languages. The ISO standard has been adapted for use in IBM host mainframes through an EBCDIC implementation, Multilingual Page (MLP) 500. This code page is directly mappable to the ISO standard.

Most personal computers, including the IBM PC, are ASCII-based. A new ASCII code page 850 has been defined and registered. This code page is directly mappable to both the ISO standard and EBCDIC MLP 500. IBM has committed to implement this new ASCII-based code page on its personal computers.

Code page 850 enables the transfer of Latin language-based data between countries using the following languages:

Belgian French	Norwegian
Canadian French	Portuguese
Danish	Spanish
Dutch	Latin-American Spanish
Finnish	Swedish
Flemish	Swiss French
French	Swiss German
German	UK English
Icelandic	US English
Italian	

In addition to improved data exchange between IBM PCs and other IBM PCs or mainframes, users of IBM PCs with code page 850 will have the same improved data exchange with non-IBM PCs and hosts that use the ISO standard code page or a mappable derivative.

Code Page Switching

Applications written for one code page may have usability problems when executed with a different code page. For example, menu boxes may turn into strings of accented characters, help text may become unreadable because alphabetics have become graphics, and so on. These problems will be avoided in future releases of IBM DOS-based operating systems by use of appropriate hardware that support multiple code pages through code page switching.

This will allow devices that support downloadable fonts, such as the EGA and Proprinter, to be switched to the appropriate code page before an application is run. Applications developed using different code pages will then run on the same IBM Personal Computer without software modification. Users will be able to access the full national language character set of code page 850 and protect their investments in applications that use other code pages.

Code page switching also has the benefit of giving the user the flexibility to develop new applications based on non-Latin or special industry based code pages.

Software developers can prepare future products for the introduction of code page 850 by:

1. Using only the 850/437 common code points in all machine-readable information (MRI).
2. Accepting as alphabetics the new accented characters available in code page 850.

Current Code Pages

Code Page 437

Figure 1 shows the standard PC code page 437. It defines 256 symbols that are all displayable, although about 30 of them are not printable on many printers because they function as printer control codes.

The lower 128 characters are based on the 7-bit ASCII code. The upper 128 characters contain characters from several European languages (including part of the Greek alphabet) and various graphic characters; however, several accented characters are missing, including some used in the Nordic countries and in Portugal. These missing characters are available in other code pages.

Many existing products, particularly those produced in the U.S., do not allow **any** characters from the upper 128 to be used.

Note that in the **registered** version of code page 437 and its variants, and in code page 850, the graphic at X'7C' is now an unbroken vertical bar instead of the broken bar. This conforms to the 7-bit ASCII standard. Although the graphic has changed, X'7C' is still the DOS pipe symbol.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				0	@	P	`	p	Ç	É	á	░	└	╨	α	≡
1			!	1	A	Q	a	q	ü	æ	í	▒	┴	╤	β	±
2			"	2	B	R	b	r	é	Æ	ó	▓	┬	╥	Γ	≥
3			#	3	C	S	c	s	â	ô	ú	│	├	╙	π	≤
4			$	4	D	T	d	t	ä	ö	ñ	┤	─	╘	Σ	⌠
5			%	5	E	U	e	u	à	ò	Ñ	╡	┼	╒	σ	⌡
6			&	6	F	V	f	v	å	û	ª	╢	╞	╓	μ	÷
7			'	7	G	W	g	w	ç	ù	º	╖	╟	╫	τ	≈
8			(	8	H	X	h	x	ê	ÿ	¿	╕	╚	╪	Φ	°
9			)	9	I	Y	i	y	ë	Ö	⌐	╣	╔	┘	θ	•
A			*	:	J	Z	j	z	è	Ü	¬	║	╩	┌	Ω	·
B			+	;	K	[	k	{	ï	¢	½	╗	╦	█	δ	√
C			,	<	L	\	l	|	î	£	¼	╝	╠	▄	∞	ⁿ
D			-	=	M	]	m	}	ì	¥	¡	╜	═	▌	φ	²
E			.	>	N	^	n	~	Ä	Pt	«	╛	╬	▐	ε	■
F			/	?	O	_	o		Å	ƒ	»	┐	╧	▀	∩	

Figure 1. Code Page 437 (Standard Version)

Code Page 850

Figure 2 shows the new code page 850. The differences between code page 850 and code page 437 are listed below by coordinates (yx): y= column, x=row. Hence, 50 is the upper case P. (Another technique of identifying a position is to use X'yx'. The meaning is the same.)

```
9B  9D  9E
A9
B5 B6 B7 B8 BD BE
C6 C7 CF
D0 D1 D2 D3 D4 D5 D6 D7 D8 DD DE
E0 E2 E3 E4 E5 E7 E8 E9 EA EB EC ED EE EF
F0 F2 F3 F4 F5 F7 F9 FB FC
```

These differences fall into five main groups:

1. Duplication of the paragraph and section symbols, (X'14' and X'15') in the region above X'80'. This makes these characters easier to print on certain printers.
2. Replacement of the single-double box graphics with accented characters.
3. Replacement of most Greek letters with accented characters.
4. Movement of the cent and yen symbols to other locations on the code page. The former location of the cent and yen signs are replaced with slashed "o"s. These Nordic characters are therefore in the same position as they are in the Nordic variants of 437.
5. A few other changes, such as the replacement of the square root symbol with a superscript "1."

Only CP437 to CP850 is compared here. Other code pages are compared to CP850 in Appendix B.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				0	@	P	`	p	Ç	É	á	░	└	ð	Ó	-
1			!	1	A	Q	a	q	ü	æ	í	▒	┴	Đ	β	±
2			"	2	B	R	b	r	é	Æ	ó	▓	┬	Ê	Ô	‗
3			#	3	C	S	c	s	â	ô	ú	│	├	Ë	Ò	¾
4			$	4	D	T	d	t	ä	ö	ñ	┤	─	È	õ	¶
5			%	5	E	U	e	u	à	ò	Ñ	Á	┼	ı	Õ	§
6			&	6	F	V	f	v	å	û	ª	Â	ã	Í	µ	÷
7			'	7	G	W	g	w	ç	ù	º	À	Ã	Î	þ	¸
8			(	8	H	X	h	x	ê	ÿ	¿	©	╚	Ï	Þ	°
9			)	9	I	Y	i	y	ë	Ö	®	╣	╔	┘	Ú	¨
A			*	:	J	Z	j	z	è	Ü	¬	║	╩	┌	Û	·
B			+	;	K	[	k	{	ï	ø	½	╗	╦	█	Ù	¹
C			,	<	L	\	l	|	î	£	¼	╝	╠	▄	ý	³
D			-	=	M	]	m	}	ì	Ø	¡	¢	═	¦	Ý	²
E			.	>	N	^	n	~	Ä	×	«	¥	╬	Ì	¯	■
F			/	?	O	_	o		Å	ƒ	»	┐	¤	▀	´	

Figure 2. Code Page 850

Character Set ID 697

A set of 190 graphic characters known as Character Set ID 697 meets the known graphic requirements of the following languages:

Belgian French	Norwegian
Canadian French	Portuguese
Danish	Spanish
Dutch	Latin-American Spanish
Finnish	Swedish
Flemish	Swiss French
French	Swiss German
German	UK English
Icelandic	US English
Italian	

These languages are used by 40 nations in Western Europe and the Western Hemisphere. All of the characters used in these languages are contained in code page 850.

This set of characters is also used in the EBCDIC Multilingual code page 500 and in all of the EBCDIC Country Extended Code Pages. This means that a host computer using a code page of Character Set 697 can interchange data with an IBM PC using CP850 with no loss of data. All of the country national characters of 40 countries are present in both machines.

The "Common Character Set"

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				0	@	P	`	p	Ç	É	á	░	└			
1			!	1	A	Q	a	q	ü	æ	í	▒	┴		β	±
2			"	2	B	R	b	r	é	Æ	ó	▓	┬			
3			#	3	C	S	c	s	â	ô	ú	│	├			
4			$	4	D	T	d	t	ä	ö	ñ	┤	─			
5			%	5	E	U	e	u	à	ò	Ñ		┼			
6			&	6	F	V	f	v	å	û	ª				μ	÷
7			'	7	G	W	g	w	ç	ù	º					
8			(	8	H	X	h	x	ê	ÿ	¿		╚			°
9			)	9	I	Y	i	y	ë	Ö		╣	╔	┘		
A			*	:	J	Z	j	z	è	Ü	¬	║	╩	┌		·
B			+	;	K	[	k	{	ï		½	╗	╦	█		
C			,	<	L	\	l	|	î	£	¼	╝	╠	▄		
D			-	=	M	]	m	}	ì		¡		═			²
E			.	>	N	^	n	~	Ä		«		╬			■
F			/	?	O	_	o		Å	ƒ	»	┐		▀		

Figure 3. Common Character Set

Writing New Applications

The first guideline is simple:

```
P-GR1   Make sure that your application will run
        on machines that are not codepage switchable.
```

Figure 4. P-GR1

If this simple guideline is broken, it will reduce the potential market for your products considerably because many existing personal computers have display adapters and printers that are not switchable. This may not matter in some new applications; however, the cost of meeting the guideline through careful design could be offset by significant benefits.

One way to reduce problems associated with multiple code pages is to limit the characters used in displaying messages on the screen to a common set of characters. This **common character set** of CP437 and CP850 is shown in Figure 3 on page 6. Wherever there is a graphic, that location has the same character in CP437 and CP850.

This common character set includes all of the lower 128 characters and 81 of the upper 128, a total of 209 characters you can use to create messages, screens, help text, etc. Users, however, should be allowed to **enter any** of the characters in column 2 through F in a code page.

Using the Common Character Set for Output Purposes

Although the graphic characters composed of mixed single and double lines are missing, it's possible to design borders around menus, tab racks for text input and frames for pop-up windows. The single-line graphic characters and the double-line graphic characters still exist and may be used for this purpose. The loss of the Greek alphabet and math symbols will affect only specialized applications.

```
P-CCS1  Restrict the graphic characters (used for boxes, etc.)
        to those in the common character set.
```

Figure 5. P-CCS1

Translation of MRI and the Common Character Set

When translating machine-readable information (MRI) such as commands, menu items and help screens, we recommend using the Common Character Set for most European languages. Unaccented forms of certain accented characters missing from the set may have to be used, but the MRI will be readable by the user regardless of the code page in effect.

When translating into Danish or Norwegian, the Common Character Set can be enlarged to include the two slashed "O" characters (X'9B' and X'9D') present in both 850 and 865 (the Nordic variant of code page 437).

For some countries, 850 is likely to become **the** standard code page almost immediately. In these countries, it may be possible (and advisable) to use the full 850 character set.

Restricting User Input

Restricting characters that the user can input may result in many significant problems. Some applications prevent users from entering the full character set because certain code points are used for control purposes. If any of these code points are alphabetics in 850, then such a program becomes restricted in a country that makes use of the unobtainable alphabetics.

```
P-ICS1  Do not reserve code points for control purposes if at
        all possible.  If you must, then make sure the code
        points you use are not alphabetic characters on any of
        the code pages.
```

Figure 6. P-ICS1

It's not just a question of enlarging the Common Character Set to include the full set of alphabetics. The user may need to use other inaccessible code points for quite legitimate reasons; for example, to imbed printer control characters. Having the displayed graphic differ, depending on the code page in effect, is irrelevant to the user who may find the loss of printer function unacceptable.

Another example concerns DOS filenames. In DOS 3.X, filenames can include many non-alphabetic characters. Restricting the input character set may mean the user is unable to access a file because its name cannot be formed with the available characters. Filenames also introduce other issues that are described in the section **File Naming Considerations**.

Folding and Monocasing

Folding is the mapping of one (usually larger) character set into another (usually smaller) one. For example, the double-line graphic characters available for display on the PC are folded into single-line graphic characters by the IBM Graphics Printer. Similarly, when text is translated from code page 437 to EBCDIC, folding is needed for symbols, such as the shading characters (X'B0', X'B1' and X'B2'), which have no EBCDIC equivalent. Folding is clearly code-page dependent.

Note: Folding implies **loss of information** and in general is not reversible.

Monocasing, another common use of folding, is the mapping of mixed- case text into upper or lower case, though the former is much more frequent. Monocasing rules are both country and code-page dependent. For example, in France the upper case form of the "e" acute has no accent, but in French-speaking Canada the upper case form has the accent.

Folding, in general, and monocasing, in particular, are often needed in PC applications:

- User input commands may be monocased to simplify searching for them in tables.
- Responses to Yes/No questions may be monocased to simplify testing.
- Monocasing of text may be provided as part of a word processor.
- Monocasing may be used to convert user input to a form suitable for passing on to DOS.
- Folding may be used as part of a sort process to get the appropriate collating sequence.

Because the rules for folding and monocasing depend not only on the code page(s) concerned but also the country, careful design is needed to ensure translation. The following guidelines should provide assistance:

P-FMC1	Isolate all folding and monocasing to a single routine with a well-defined interface.
P-FMC2	Design the routine so that it is entirely table-driven; it then can be adapted easily for use with another code page or in another country.
P-FMC3	Make the tables accessible to the user so they can be over-ridden, if necessary.
P-FMC4	Use DOS facilities (e.g. function call 38H) to obtain the information needed to build the default table because DOS is aware of both the code page and country.

Figure 7. P-FMC1, 2, 3, 4

For more information about collating sequence issues, see **National Language Support Items** on page 14.

Words and Text Delimiters

Any program that deals with text has to make assumptions about the delimiters that separate words from each other. This can be done in two ways:

1. Use a table of valid delimiters and treat anything that is not a delimiter as part of a word.
2. Use a table of valid alphabetics and treat anything that is not alphabetic as a delimiter.

Because there is a whole set of new alphabetics in CP850, the table used in the second approach will be code-page dependent.

P-WDL1	Isolate words by using a table of valid delimiters that is not code-page specific. Avoid using a table of valid alphabetics because it will depend on the code page in use.

Figure 8. P-WDL1

Dealing with Data Files

In the planned implementation of DOS 3.30 and Operating System/2™ there is no way for an application to tell which code page was used to encode a data file. While this may not matter for many applications, there are situations in which major damage can be caused by a wrong assumption.

The most obvious example is a data base manager. Because 437 does not contain all the alphabetics in 850, attempts by a user in 437 mode to create or update records in a data base currently encoded using 850 can lead to data corruption. First, he will be unable to enter certain accented characters and will use unaccented versions instead (which may mean that later searches for the data will fail); second, he will see some existing data as garbage and may attempt to "correct" it.

Where corruption of this sort is viewed as a major hazard, it is suggested that the application record, in each data file, the code page that was in effect when the file was created. This "tag" can then be checked whenever the file is accessed to see if it matches the current code page. If it does not, the application can:

- Inform the user of the mismatch and terminate, or
- Offer the user the option of switching to the right code page and then continuing.

```
P-TAG1  If the risk of data corruption through a code-page
        mismatch is high, maintain a "code page in effect when
        created" tag for each data file.
```

Figure 9. P-TAG1

Data Conversion

There are several ways to make a new application, designed to run in one code page environment, able to handle data files created using another:

- A separate utility program could be made available to allow a one-time translation of the data.
- The application could switch code pages dynamically.
- The application could convert the data dynamically.

A "smart" application which implements a file tag can automatically detect that some sort of data conversion is needed. Even if the application does not maintain such a tag, the user must still be able to cope with the problem.

Conversion Utilities

The use of conversion utilities offers several advantages:

1. It relieves the main application from having to worry about dynamic data conversion or code page switching.
2. The main application will be smaller, simpler and probably faster.
3. It removes the multiple window problem (see **Dynamic Code Page Switching Within an Application**).
4. By doing the conversion as a separate, stand-alone step, it can be made user-friendly, even interactive, without adding to the complexity of the main application.

A conversion utility has to know quite a bit about the file structure it is dealing with. It certainly isn't possible to translate all X'9D' bytes to X'BE' just because the yen symbol is X'9D' in 437 and X'BE' in 850. Some X'9D' bytes may represent field lengths or attribute bytes, or they may be part of a binary number. Thus, a utility that converts a spreadsheet DIF file is not likely to do a good job of converting a word processing document and vice versa.

If your product has a unique file structure, then you should provide such a utility. This allows the user to reconsider the decision about which code page to use. The alternative for him is to keep the data in the same code page forever (which may prevent making it available to others on a LAN, for example) or to write a conversion utility. In either case, the usability and marketability of the product will suffer.

```
P-CNV1  Provide conversion utilities if your product has its
        own unique file structure.
```

Figure 10. P-CNV1

Note that a conversion utility usually has to do some folding. For example, the mixed single-double line graphics has to be converted to either the single-line or double-line graphics when going from 437 to 850, while accents have to be removed from some characters in going from 850 to 437. To retain flexibility and give the user some control over such matters, use the following guideline.

```
P-CNV2  Since there is no perfect mapping from 437 to 850, make
        any conversion utility table-driven, and make the
        tables accessible to the user.
```

Figure 11. P-CNV2

Where the application maintains a file tag, it should, of course, check to see if the file has already been converted.

Dynamic Code Page Switching Within An Application

Although there are the necessary DOS commands and application programming interface for dynamic code page switching, a number of things must be considered. Unless the application maintains a private file tag, only the user will know if the switching is necessary; it can't be done automatically.

Another consideration arises when an application allows multiple windows into separate data files. Since the current code page is applied across the whole screen, it is not possible to display one window in 437 mode and another in 850 mode. This affects many windowing products and split screen editors. The situation can also arise if P-CCS1 has not been followed (see Figure 5 on page 7). In such a case, the application's menu, help text or messages may contain characters that are specific to code page 850 while the data has 437-specific characters in it.

P-DCP1	Don't dynamically switch code pages in a product that has the ability to display data from more than one file at the same time.

Figure 12. P-DSP1

Even if only one file may be viewed at a time, it still may be inappropriate to perform code page switching if the product allows or encourages rapid switching between files. Some editors provide the ability to move very quickly around a ring of currently edited files.

Another item to consider is the effect of dynamic code page switching and printing. Switching the printer while it is printing a document (e.g., because a background print spooler is being used) should be avoided. If switching is necessary, wait until the printer has completed the print job.

P-DCP2	If dynamic code page switching is used, make sure that no problems are caused by switching the printer and keyboard at the same time as the display.
P-DCP3	If dynamic code page switching is used, use the new command CHCP. Do not use the new parameters added to the MODE command in DOS 3.3.

Figure 13. P-DCP2 and P-DCP3

Applications that allow the user to shell out to DOS face another problem: the user could accidentally, or deliberately, invoke a command or program that would change the current code page. This could cause problems when control returns to the original application. To avoid such problems, an application that permits shelling out should:

- Remember the current code page before shelling out
- Test the code page when control returns
- Switch back if the code page has been altered

This is one of the few situations in which dynamic code page switching is recommended.

Finally, when designing applications for dynamic switching, don't forget P-GR1! (See Figure 4 on page 7.) Make sure your product works, even if in a degraded fashion, on a machine that has no switching capability.

Dynamic Data Conversion Within An Application

An application that allows dynamic data conversion is not recommended. Unless the user is able to reverse the process (convert the data back to 437 form), other applications that need the data in 437 form will not be able to process it. Conversion implies folding and folding, in general, loses information, so the double conversion also will cause problems.

P-DDC1	Don't provide dynamic data conversion in an application.

Figure 14. P-DDC1

A one-time conversion using a separate utility is much more sensible.

Data Sharing

There are a number of ways in which data files may be shared between users:

- simple exchange of diskettes
- via a local area network
- via a communications link (asynch, 3270 emulation, etc.)

The opportunities for data corruption can be avoided only by making sure that all files use the same code page unless:

- The files concerned are tagged in some way

- All the software concerned takes account of, and preserves the tags
- Utilities exist for converting the files between the different code pages

Implementing a File Code-Page Tag

As mentioned earlier, a file tag is useful in preventing a user from corrupting data and in determining when a code page switch would be useful. The information should be accessed through a single module to simplify maintenance and to allow the same code to be used in many applications.

```
P-CPT1  The code to open a file for input should be designed to
        retrieve the "code page in effect when created" tag.
        This information should be accessible to the rest of
        the program using the routine in a simple, standard
        way.
```

Figure 15. P-CPT1

File Naming Considerations

DOS currently folds filenames to upper case in a "437" way. The rules it follows are a compromise between the various country conventions. An X'82' (lower case e acute) is folded to X'45' (upper case E), while X'9B' (cent) stays as X'9B'. However, in code page 850, X'9B' is a lower case slashed O which should be folded to an upper case slashed O, X'9D'. Any filename folding imbedded in a product must be changed to cope with 850. Where possible, bypass the problem altogether, at least as far as existing files are concerned.

```
P-FNM1  Where possible, let the user select file names from a
        scrollable directory, rather than entering them through
        the keyboard.  Such a directory should be sortable by
        name, date, size, etc.
```

Figure 16. P-FNM1

Where this approach is not possible, then:

```
P-FNM2  Let DOS perform any folding.  This ensures that the
        right folding rules will be applied.
```

Figure 17. P-FNM2

Implementation Considerations

The architecture of the IBM PC dictates the way in which DOS implements code-page switching support. Any application that goes below the DOS function call interface must be tested carefully to ensure that it does not interfere with the keyboard, video and print interrupt mechanisms which provide that support.

For example, the display support (limited to text modes) involves the use of downloaded fonts. Applications that exploit EGA capabilities can interfere with the code-page switching mechanism:

- The EGA provides some special modes, (such as the 43-line mode exploited by IBM Personal Editor Version 2) that do not have code page switching support because they use fixed, hardware-resident fonts. Switching to these modes may temporarily disable code page switching.
- Any application that uses the additional INT 10H functions provided by the EGA must ensure that it does not interfere with the fonts loaded by DOS.
- Any application that hooks into the video interrupt, INT 10H, must ensure that it does not interfere with the DOS code page switching mechanism.

Similar considerations apply to applications that exploit the capabilities of the IBM ProPrinter or that hook into the keyboard interrupt (INT 9H) or the printer interrupts (INT 5H and INT 17H).

Text in Graphics Modes

When a CGA or EGA is in a graphics mode, any text displayed on the screen is generated in a different way from text displayed in a non-graphics mode. Some products, particularly those with high graphics function, contain their own text fonts for use in this situation.

Since these fonts are an internal part of the product, they must be altered when code page switching occurs.

```
P-TGF1  Design any additional text support for graphics mode to
        include all the new characters in code page 850.
```

Figure 18. P-TGF1

Since 75 percent of the font is unchanged when code page switching occurs, it is sensible to switch only those characters that need altering.

```
P-TGF2  Design the font switching so that a subset of the whole
        character set can be altered.
```

Figure 19. P-TGF2

Capturing Screen Images

A few specialized applications have the ability to capture a screen image. A problem arises here when a text screen is captured and one code page is active and redisplayed while another is in effect. For example, a screen captured in 437 mode that contains X'9D', the yen symbol, will show a capital slashed "O" if redisplayed in 850 mode. Of course, there is a yen symbol in 850, but it is X'BE'.

There isn't a complete solution to this. For example, code page switching is done at the screen level, so it's not possible to include in a 437 screen some picture element that was captured and contained 850-specific characters. Even avoiding the problem requires finding out which code page was in effect when the screen was originally captured.

Host Connection and EBCDIC/ASCII Translation

Unlike code page 437 and its variants, code page 850 provides a one-for-one mapping of alphabetics between it and the Character Set ID 697. If your product incorporates an emulator of its own, make sure that:

```
P-TEM1  The EBCDIC/ASCII mappings use ASCII MLP 850 and EBCDIC
        MLP 500.
```

Figure 20. P-TEM1

At the time this was written, there were no registered standard mappings, but work is under way to provide at least a recommended set. In the interim, there is a way to cope with problems that may arise with applications that have not been fully converted.

```
P-TEM2  Ensure that all mapping is done through tables that are
        accessible to the user.
```

Figure 21. P-TEM2

When translating EBCDIC host data to ASCII on the personal computer, there is usually little risk and considerable benefits when the default translation is to 850 rather than 437. There is little risk because in most countries the 3270 keyboards generally available provide only part of the common subset. The benefits will come later when 850 users will want to upload 850 data to the host.

Round-Trip Considerations

Where files are uploaded from PC and host with ASCII/EBCDIC translation, it may be important that the data be identical to the original when it is downloaded. This round-trip requirement means that any mapping tables should be reversible.

Changing Existing Applications

It is much easier to design a new application **for** code page switching than to design code page switching **into** an existing product. The following checklist is intended to help you determine how much an existing application will be impacted by Code Page 850. It identifies where problems could occur, and suggests ways to circumvent or minimize them.

1. Does the product allow or use code points above X'7F'?

   If it does, then it will almost certainly require some modification to run in a code page switching environment. Programs that use only the lower 128 characters will not be affected, although they will be limited in usefulness outside the USA.

2. Does the product contain its own fonts for use in graphics modes?

   If so, new fonts or partial fonts will be needed for the new characters. See Figure 18 on page 11 and Figure 19 on page 12.

3. Does the product contain code to fold character strings to upper case?

   If so, this will need amending to deal with the new alphabetic characters. Note that this is actually both a code page and a country-dependent issue. As an alternative, you might plan to use the monocasing routine accessible via function call 38H in DOS. See Figure 7 on page 8.

4. Does the product contain code to convert between ASCII and EBCDIC?

   If so, the conversion tables will need updating. See Figure 20 on page 12 and Figure 21 on page 12.

5. Does the product use the mixed single-double line graphic characters to draw boxes, windows or frames?

   If so, it will need to be amended to use either the set of single- or double-line graphics. See Figure 5 on page 7.

6. Does the product make assumptions about or contain its own collating sequence?

   If so, this will need amending to cope with the new upper and lower case alphabetic characters.

7. Does the product use a filter to prevent the user from entering certain characters because the corresponding code points are used for control information?

   If these code points are used for control information, reassign any that represent alphabetic characters in CP850. See Figure 6 on page 7.

8. Does the product parse or filter filenames that are entered by the user before passing them to DOS?

   If so, check that the parsing or filtering is still appropriate. For example, if the product folds filenames to upper case, there may be problems. See Figure 16 on page 11 and Figure 17 on page 11.

9. Does the product make assumptions about what can delimit a word of text by using a table of valid alphabetics?

   If so, this table will be code-page dependent. Consider switching to a table of valid delimiters. See Figure 8 on page 8.

10. Does the product exploit the capabilities of the EGA or ProPrinter in ways that interfere with the implementation of code page switching?

    If so, rewriting may be necessary.

11. Does the product hook into the keyboard, display or printer interrupt vectors (9H, 16H, 17H and 05H)?

    If so, care must be taken that this does not interfere with the implementation of code page switching.

National Language Support Items

The document "Software Without Frontiers" (GE19-5356) contains a great deal of information about developing software for the international market so that adaptation and translation are not painful and expensive tasks. This section identifies a few of the more important items.

The treatment of Machine-Readable Information (MRI) is a very important part. MRI (the text of menus, prompts, messages, commands, report headings, etc) will need to be translated, and this task will be enormously simplified if:

- MRI is isolated from the executable code.

  It is not only packaged separately but loaded separately, for best results.
- Sufficient space is made available for expansion when MRI is translated. This is language-dependent, but more important, it also depends on the text length. A 10-character message can expand by 150 percent or more, while a 250 character message is unlikely to expand by much more than 30 percent.
- Messages are complete entities and not constructed out of a common set of words or phrases. This avoids problems with word order in different languages.

Another area to be considered is the format of such items as dates, times, and currencies that differ from country to country. Make sure these formats are selectable. The DOS International Call programming interface provides most of the information required (function 38H for DOS version 3.0 and later).

Collating and sorting require considerable effort if they are to be done properly. For example:

- Folding and monocasing are country-dependent
- In some languages, double or triple letter sequences should be treated as a single character. Thus, in Catalan, the word "llama" should come before the word "lana" because the double "ll" is treated as a single "l."
- The user should be able to specify that certain character sequences are to be ignored. For example, when surnames are being sorted, prefixes like "van der" or "d" are not used when collating names in telephone directories.

List of Principles

P-GR1	Make sure that your application will run on machines that are not codepage switchable.
P-CCS1	Restrict the graphic characters (used for boxing, etc.) to those in the common character set.
P-ICS1	Do not reserve code points for control purposes, if at all possible. If you must, then make sure the code points you use are not alphabetic characters on any of the code pages.
P-FMC1	Isolate all folding and monocasing to a single routine with a well-defined interface.
P-FMC2	Design the routine so that it is entirely table-driven; it then can be easily adapted for use with another code page or in another country.
P-FMC3	Make the tables accessible to the user so that they can be overridden, if necessary.
P-FMC4	Use DOS facilities (e.g., function call 38H) to obtain the information needed to build the default table because DOS is aware of both the code page and country.
P-WDL1	Isolate words by using a table of valid delimiters that will not be code-page specific. Avoid using a table of valid alphabetics because it will depend on the code page in use.
P-TAG1	If the risk of data corruption through a code-page mismatch is high, maintain a "Code-Page in effect when created" tag for each data file.
P-CNV1	Provide conversion utilities if your product has a unique file structure.
P-CNV2	Since there is no perfect mapping from 437 to 850, make any conversion utility table-driven, and make the tables accessible to the user.
P-DCP1	Don't dynamically switch code pages in a product that has the ability to display data from more than one file at the same time.
P-DCP2	If dynamic code page switching is used, make sure that no problems are caused by switching the printer and keyboard at the same time as the display.
P-DCP3	If dynamic code page switching is used, use the new command CHCP. Do not use the new parameters added to the MODE command in DOS 3.30.
P-DDC1	Don't provide dynamic data conversion in an application.
P-CPT1	The code to open a file for input should be designed to retrieve the "code page in effect when created" tag. This information should be accessible to the rest of the program using the routine in a simple, standard way.
P-FNM1	Where possible, let the user select file names from a scrollable directory rather than entering them through the keyboard. Such a directory should be sortable by name, date, size, etc.
P-FNM2	Where possible, let DOS perform any folding. This ensures that the right folding rules will be applied.
P-TFG1	Design any additional text support for graphics mode to include all the new characters in code page 850.
P-TFG2	Design the font switching so that a subset of the whole character set can be altered.
P-TEM1	The EBCDIC/ASCII mappings use ASCII MLP 850 and EBCDIC MLP 500.
P-TEM2	Ensure that all mapping is done through tables that are accessible to the user.

Appendix A

Code Page 850 To 437 Comparisons

The chart identifies locations where characters differ (column row indicators) on code page 850 and code page 437.

Avoid using characters represented at those locations in any information presented to the user on the screen (MRI). The resulting application will run on a PC with code page 850 or code page 437; i.e., it will be independent of code page 437 and 850.

An alternative use of these comparisons is to determine which characters in a 437-based application are impacted when an application is modified to execute on an 850-based PC also.

Because a number of the characters on these code pages do not print on many printers in the U.S., the character ID is used in the table instead of the character itself.

HEX VALUE	C.P. 850 CHARACTER	C.P. 437 CHARACTER
9B	LO61	SC04
9D	LO62	SC05
9E	SA07	SC06
A9	SM53	SM68
B5	LA12	SF19
B6	LA16	SF20
B7	LA14	SF21
B8	SM52	SF22
BD	SC04	SF27
BE	SC05	SF28
C6	LA19	SF36
C7	LA20	AF37
CF	SC01	SF45
D0	LD63	SF46
D1	LD62	SF47
D2	LE16	SF48
D3	LE18	SF49
D4	LE14	SF50
D5	LI61	SF51
D6	LI12	SF52
D7	LI16	SF53
D8	LI18	SF54
DD	SM65	SF58
DE	LI14	SF59
E0	LO12	GA01
E2	LO16	GG02
E3	LO14	GP01
E4	LO19	GS02
E5	LO20	GS01
E7	LT63	GT01
E8	LT64	GF02
E9	LU12	GT62
EA	LU16	GO32
EB	LU14	GD01
EC	LY11	SA45
ED	LY12	GF010001
EE	SM15	GE01
EF	SD11	SA38
F0	SP32	SA48
F2	SM10	SA53
F3	NF05	SA52
F4	SM25	SS26
F5	SM24	SS27
F7	SD41	SA70
F9	SD17	SA79
FB	ND011	SA80
FC	ND031	LN011

Appendix B

Differences Between Code Page 850 and Other Code Pages

The following charts identify differences between code page 850 and other PC code pages. Use these charts to create code page-independent applications. For example, to create an application independent of code pages 850, 437 and 860, do not use locations that have character ID's in the columns marked 437 or 860.

Note: The charts use code page 850 as the base code page.

HEX VALUE	C.P.850 MLP	C.P.437 DEFAULT	C.P.860 PORTUGAL	C.P.861 ICELAND	C.P.863 CAN FREN	C.P. 865 NORDIC-2	C.P. 437N NORDIC
84	LA17		LA19		LA16		
86	LA27		LA12		SM25		
89	LE17		LE16				
8B	LI17		LI12	LD62			
8C	LI15		L016	LD63			
8D	LI13			LT64	SM10		
8E	LA18		LA20		LA14		
8F	LA28		LA16		SM24		
91	LA51		LA14		LE14		
92	LA52		LE14		LE16		
94	L017		L019		LE18		
95	L013			LT63	LI18		
96	LU15		LU12				
97	LU13			LY12			
98	LY17		LI14	LY11	SC01		
99	L018		L020		L016		
9B	L061	SC04	SC04		SC04		
9D	L062	SC05	LU14		LU14		
9E	SA07	SC06	SC06	SC06	LU16	SC06	LL64
9F	SC07		L012				LL63
A0	LA11				SM65		
A1	LI11				SD11		
A4	LN19			LA12	SD17		
A5	LN20			LI12	SD41		
A6	SM21			L012	ND031		L019
A7	SM20			LU12	SM15		L020
A8	SP16				LI16		

HEX VALUE	C.P.850 MLP	C.P.437 DEFAULT	C.P. 860 PORTUGAL	C.P.861 ICELAND	C.P. 863 CAN FREN	C.P. 865 NORDIC-2	C.P.437N NORDIC
A9	SM53	SM68	LO14	SM68	SM68		LA19
AA	SM66						LA20
AB	NF01						SM16
AC	NF04						LN63
AD	SP03				NF05		
AE	SP17						ND031
AF	SP18					SC01	SC01
B5	LA12	SF19	SF19	SF19	SF19	SF19	SF19
B6	LA16	SF20	SF20	SF20	SF20	SF20	SF20
B7	LA14	SF21	SF21	SF21	SF21	SF21	SF21
B8	SM52	SF22	SF22	SF22	SF22	SF22	SF22
BD	SC04	SF27	SF27	SF27	SF27	SF27	SF27
BE	SC05	SF28	SF28	SF28	SF28	SF28	SF28
C6	LA19	SF36	SF36	SF36	SF36	SF36	SF36
C7	LA20	SF37	SF37	SF37	SF37	SF37	SF37
CF	SC01	SF45	SF45	SF45	SF45	SF45	SF45
D0	LD63	SF46	SF46	SF46	SF46	SF46	SF46
D1	LD62	SF47	SF47	SF47	SF47	SF47	SF47
D2	LE16	SF48	SF48	SF48	SF48	SF48	SF48
D3	LE18	SF49	SF49	SF49	SF49	SF49	SF49
D4	LE14	SF50	SF50	SF50	SF50	SF50	SF50
D5	LI61	SF51	SF51	SF51	SF51	SF51	SF51
D6	LI12	SF52	SF52	SF52	SF52	SF52	SF52
D7	LI16	SF53	SF53	SF53	SF53	SF53	SF53
D8	LI18	SF54	SF54	SF54	SF54	SF54	SF54
DD	SM65	SF58	SF58	SF58	SF58	SF58	SF58
DE	LI14	SF59	SF59	SF59	SF59	SF59	SF59
E0	LO12	GA01	GA01	GA01	GA01	GA01	GA01
E2	LO16	GG02	GG02	GG02	GG02	GG02	GG02
E3	LO14	GP01	GP01	GP01	GP01	GP01	GP01
E4	LO19	GS02	GS02	GS02	GS02	GS02	GS02
E5	LO20	GS01	GS01	GS01	GS01	GS01	GS01
E7	LT63	GT01	GT01	GT01	GT01	GT01	GT01
E8	LT64	GF01	GF01	GF01	GF01	GF01	GF01
E9	LU12	GT62	GT62	GT62	GT62	GT62	GT62
EA	LU16	GO32	GO32	GO32	GO32	GO32	GO32
EB	LU14	GD01	GD01	GD01	GD01	GD01	GD01
EC	LY11	SA45	SA45	SA45	SA45	SA45	SA45
ED	LY12	GD010001	GD010001	GD010001	GD010001	GD010001	GD010001
EE	SM15	GE01	GE01	GE01	GE01	GE01	GE01
EF	SD11	SA38	SA38	SA38	SA38	SA38	SA38
F0	SP32	SA48	SA48	SA48	SA48	SA48	SA48
F2	SM10	SA53	SA53	SA53	SA53	SA53	SA53
F3	NF05	SA52	SA52	SA52	SA52	SA52	SA52
F4	SM25	SS26	SS26	SS26	SS26	SS26	SS26
F5	SM24	SS27	SS27	SS27	SS27	SS27	SS27
F7	SD41	SA70	SS70	SA70	SA70	SA70	SA70
F9	SD17	SA79	SA79	SA79	SA79	SA79	SA79
FB	ND011	SA80	SA80	SA80	SA80	SA80	SA80
FC	ND031	LN011	LN011	LN011	LN011	LN011	LN011

IBM Personal System/2 Seminar Proceedings

Publication Number	Vol.	Topic
G360-2653	V5.1	IBM Personal System/2 Model 30 IBM Personal Computer DOS, Version 3.30
G360-2678	V5.2	IBM Personal System/2 Displays and Display Adapters
G360-2637	V5.3	IBM Personal System/2 Models 50, 60, 80 Micro Channel™ Architecture, Hardware Features and Design Considerations
G360-2747	V5.4	IBM Personal System/2 Models 50, 60, 80 VGA, BIOS and Programming Considerations
G360-2756	V5.5	IBM Operating System/2
G360-2758	V5.6	IBM Personal Computer Multilingual Code Page 850 Application Development Considerations

Notes

G360-2758

IBM Corporation
Editor, IBM Personal System/2 Seminar Proceedings
Internal Zip 3636
Post Office Box 1328
Boca Raton, Fl 33429-1328