Assessment of Options for Handling Full Unicode Character Encodings in MARC21
A Study for the Library of Congress

Part 1: New Scripts

Jack Cain
Senior Consultant
Trylus Computing, Toronto

1 Purpose

This assessment intends to study the issues and make recommendations on the possible expansion of the character set repertoire for bibliographic records in MARC21 format.

1.1 "Encoding Scheme" vs. "Repertoire"

An encoding scheme contains codes by which characters are represented in computer memory. These codes are organized according to a certain methodology called an encoding scheme. The list of all characters so encoded is referred to as the "repertoire" of characters in the given encoding schemes. For example, ASCII is one encoding scheme, perhaps the one best known to the average non-technical person in North America. "A", "B", & "C" are three characters in the repertoire of this encoding scheme. These three characters are assigned encodings 41, 42 & 43 in ASCII (expressed here in hexadecimal).

1.2 MARC8

"MARC8" is the term commonly used to refer both to the encoding scheme and its repertoire as used in MARC records up to 1998. The '8' refers to the fact that, unlike Unicode which is a multi-byte per character code set, the MARC8 encoding scheme is principally made up of multiple one byte tables in which each character is encoded using a single 8 bit byte. (It also includes the EACC set which actually uses fixed length 3 bytes per character.) (For details on MARC8 and its specifications see: http://www.loc.gov/marc/.)

MARC8 was introduced around 1968 and was initially limited to essentially Latin script only. Gradually it was expanded until today it includes the following scripts: Arabic, Chinese, Cyrillic, Greek, Japanese, Korean, and Latin. The vast majority of bibliographic records in North America and in many other locations around the world are exchanged using MARC8 encoding.

Very little expansion has been made to the MARC8 repertoire in recent years, as a decision was made in the early 1990s to look toward Unicode for additional characters rather than continue the arduous task of expanding MARC8. (The term 'arduous' is deliberate and refers not so much to the difficulty of making additions and documenting them but to the labor that every computer systems vendor using MARC records must expend in order to modify their systems so that new characters are recognized and

supported.) As the users and systems have proliferated, change has become more costly to the community at large.

Although MARC8 is based on ASCII, parts of the repertoire and encodings outside of ASCII are unique to the world of libraries and library records and have been little used outside this domain causing support challenges. Therefore the concept of the adoption of Unicode also has the attraction of bringing the library world more into line with mainstream computer developments such as the Internet.

1.3 Unicode

The Unicode encoding scheme and its repertoire have been in development for a little over 10 years now. The intent of Unicode is to provide a single encoding scheme that is capable of handling all the world's languages. Although its adoption has not been as quick as was initially hoped or predicted, Unicode is now the underlying encoding used in many major software development efforts—all recent Microsoft products, the Java programming language, and so on.

In the year 1998, it was agreed [MARBI Proposal 98-18], that it was acceptable for MARC21 libraries participating in data interchange to begin using the Unicode encoding scheme as an alternative to the MARC8 encoding scheme. However, it was also agreed that the character set repertoire in current use was not to be expanded. The prohibition on repertoire expansion was considered necessary because of concerns over record exchange among systems—a vital element in the world of library information processing.

This MARBI decision then meant that MARC21 records could be encoded in either MARC8 or in Unicode but that only the MARC8 repertoire of characters was to be allowed in either case. The MARC8 repertoire represents all of the characters in the MARC8 encoding scheme but it represents only a small fraction of all the characters in the Unicode encoding scheme.

1.4 Issues

There is some urgency in the need to come to agreement on the resolution of the issues being raised in this report since already some local library systems are running on Unicode and several more are in the process of developing Unicode-based systems. Although users of these systems can be encouraged to stay within the current MARC8 repertoire, the systems require specialized software filters designed to ensure that no characters outside the current MARC8 repertoire enter the system. If such filters are not provided, some characters outside the current MARC8 repertoire will begin to appear on such systems and will then begin to find their way into distribution channels and subsequently appear in records destined to be loaded on non-Unicode systems.

Canadian Aboriginal Syllabics. MARBI proposal 2002-11 allowed the addition of Canadian Aboriginal Syllabics to the MARC21 repertoire but only in Unicode encoding—thus obviating the need to expand the current MARC8 repertoire but recognizing that there is a loss of data if a conversion to MARC8 is needed for an interchange situation.

Communication Format vs Internal Processing. In considering the various issues raised in this report, systems engineers need to decide what is a matter of internal handling within the system and what is a matter for inclusion within records being issued in MARC communication format. The stability of the repertoire and encodings in MARC8 have contributed greatly to record sharing, cooperative projects, and vendor system development - which have brought cost savings to libraries. Therefore the exchange environment is the primary issue, rather than the internal system, although the two can be more cost effective when compatible.

1.5 Basic Multilingual Plane (BMP)

The present study is limited to the Basic Multilingual Plane of Unicode.

> "The Basic Multilingual Plane (BMP, or Plane 0) contains all the common-use characters for all the modern scripts of the world, as well as many historical and rare characters. By far the majority of all Unicode characters for almost all textual data can be found in the BMP." (The Unicode Consortium. *The Unicode standard, version 4.0*. Reading, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1, page 35)

It should be noted that access to characters beyond the BMP requires special techniques as specified in the Unicode documentation. System software must be aware of such techniques and library automation systems must take them into account in their design for characters beyond the BMP to be available to the applications running on them. While it is very likely that most library systems would only extremely rarely need to access characters beyond the BMP, it should be noted that there are already a couple of han (Chinese) characters in the current MARC8 repertoire which fall outside the BMP in Unicode. The Unicode standard, version 4.0, has this to say about Plane 2 where these han characters have been placed:

> "…the vast majority of Plane 2 characters are extremely rare or of historical interest only".

2 New Scripts and New Characters in Existing Scripts

The move from the current MARC8 repertoire to a "full Unicode" repertoire represents an enormous increase in the number of characters to be handled. The current MARC8 repertoire includes about 17,000 characters. Unicode 4.0 includes 236,029 code points of which 50,635 are for graphic characters in the "Basic Multilingual Plane". (Version 4.0 of Unicode was released in the fall of 2003.). There are two areas of concern. One is for new characters not in the MARC8 repertoire. Another is for characters that are in the repertoire but have alternative encodings in Unicode. This section is focused on the former.

The Unicode encoding 'code space' is divided into 'blocks' or ranges of code points. "Appendix B: Unicode 4.0 Blocks and MARC8 Encoding" provides a list of all Unicode blocks and shows which blocks have partial coverage in the MARC8 repertoire and which are entirely new.

"Appendix A: Unicode Scripts" provides a similar listing but is ordered by script and shows the number of characters in Unicode that are in addition to those found in the MARC8 repertoire. A further set of tables giving specific code points in each Unicode block that are new to the MARC8 repertoire has been prepared as an ancillary part of this report.

2.1 Moving from Full Unicode Records to MARC8

The issue that raises the most concern in repertoire expansion is the complex of consequences encountered when records move from a system with a large repertoire to a system that has provision only for a much smaller repertoire. Character set is very fundamental and very unforgiving—if a system does not know about a character code, or worse—has a different character assigned already to that code—then data corruption will result unless proper measures are taken.

A number of options are available when moving data from a full Unicode environment to the current MARC8 repertoire environment. In making a choice of option, it should first of all be stated that the move to full Unicode systems is likely to be an irreversible trend with more and more systems moving in this direction and with therefore more and more pressure for others to do the same. With this in mind, it does not seem reasonable to invest in very complex solutions.

2.1.1 Option 1: Drop the character

In this option the characters that are found not to be present in the current MARC8 repertoire are deleted. The software used to perform the deletion also must be capable of handling all resultant changes to the record arising from the deletion. The following structural checking needs to be applied before the record is stored in the system in question and before any indexing is applied to it.

a) if the character in question is a precomposed character it may be preferable to decompose it first and then analyze what needs to be done next. If both the character and its diacritic can be kept in MARC8 then both should be kept; if the diacritic is not present in MARC8 then at least the letter should be kept. If neither is in MARC 8 then both will have to be dropped.
b) make any necessary adjustments in the length calculations recorded in the record
c) if, after such deletions, an empty field or subfield remains then that field or subfield would need to be deleted

2.1.2 Option 2: Substitute another character

A list of sensible substitution characters may be developed to handle commonly encountered characters for which reasonable substitution characters exist. For example in

Unicode the ellipsis has an encoding as a single character [U+2026], but in MARC 8 three ASCII periods could be used as a substitute.

The basic advantage of this method is its relative simplicity and practicality. The basic disadvantage is the loss of information about the original character although it is true that some such substitutions could be reversed. A second disadvantage is that this solution would only be practical for a limited number of characters for which a reasonable substitution exists. For the overwhelming majority of characters in Unicode that are beyond the MARC8 repertoire no sensible substitution is likely to be found.

It should also be stated that character substitution is against Unicode principles since it destroys data integrity—although this conformance applies to Unicode systems only and if the data is being moved to a non-Unicode system there is some justification for not being constrained by this conformance requirement. (See Unicode Conformance Requirement, C10. Unicode 4.0, section 3.2, "Modification", page 60)

2.1.3 Option 3: Replace the character with a "place-holder" character

(Cf. Unicode section 5.3: Unknown and Missing Characters)
This option is similar to Option 2 but instead of deleting the character, it is replaced by a "place-holder" character. This technique is often encountered on the web where the empty rectangle is commonly used as a place-holder character for a character that is not present. It should be noted that the concept of a place-holder character has existed for some time in CJK character sets where a specific character for this purpose has been assigned and is present in the current MARC8 repertoire. This character, called "geta", has hexadecimal code 212A46 in the current MARC8 encoding and is encoded as U+0313 in Unicode.

If this option is chosen, the simplest implementation would be to use the same place-holder character for all characters missing from the current MARC8 repertoire. One might also want to consider a combination of options 2 and 3: substitution for some characters; place-holders for others. One of the consequences of the adoption of this technique is that a new place-holder character would have to be established in MARC8 for non-CJK data—probably in one of the spare code points in the ANSEL code space, such as C9. The obvious disadvantage of this option is the loss of information about the original character.

2.1.4 Option 4: Replace the character with "[U+nnnn]" in ASCII

In this option the character or characters in question are replaced by a conventional representation in ASCII characters of the 2 byte Unicode hexadecimal code point for the character in question. For example, if the original Unicode record contains an ellipsis, a character not in the current MARC8 repertoire, the character would be replaced by "[U+2026]".

This method requires the software to make any necessary adjustments in the length calculations recorded in the record. It should also be noted that this technique could cause an "overflow" problem with records or fields that are long since every character so

treated would increase by a number of bytes. The severity of the problem would depend on the limits imposed by the software within which the records are handled.

This method has the advantage that the value of the original character has not been entirely discarded and still may be reconstructed from the code value that is recorded. Performing such a reconstruction however is not a trivial task.

2.1.5 Option 5: Replace the non-MARC8 character with the Unicode character name

This option, suggested by Joan Aliprand of RLG, follows the "tried and true" method of providing a cataloger's description for characters that the technology cannot handle. It is essentially the same as Option 4 above but uses the more "user-friendly" approach of a readable name instead of a numeric code point. The Unicode character name is extractable from the Unicode Character Database. (See 'Resources' below)

Note that it is not helpful with han characters which essentially do not have a name in Unicode. And the possible length problems discussed in point 4 are even more severe with this method.

2.1.6 Option 6: Using escape sequences to shift to Unicode within records encoded in the current MARC8 encoding

Although this option is theoretically possible it is strongly discouraged by this report. Such a course of action will inevitably make for severe complications in systems design and therefore is not to be recommended. The following quotation from the Unicode 4.0 text provides good background on this point.

> "The Unicode standard, by supplying a universal repertoire associated with well-defined character semantics, does not require the *code set independent* model of internationalization and text handling. That model abstracts away string handling as manipulation of byte streams of unknown semantics to protect implementations from the details of hundreds of different character encodings, and selectively late-binds locale-specific character properties to characters. Of course, it is always possible for code set independent implementations to retain their model and to treat Unicode characters as just another character set in that context. It is not at all unusual for Unix implementations to simply add UTF-8 as another character set, parallel to all the other character sets they support. However, by contrast, the Unicode approach—because it is associated with a universal repertoire—assumes that characters and their properties are inherently and inextricably associated. If an internationalized application can be structured to work directly in terms of Unicode characters, all levels of the implementation can reliably and efficiently access character storage and be assured of the universal applicability of character property semantics." (The Unicode Consortium. *The Unicode standard, version 4.0*. Reading, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1, page 15)

2.1.7 Recommendations

The choice of which of the above methods is influenced by a number of factors:
a) availability of software to perform the conversion.
b) cost of conversion software, especially if it must be created from scratch.
c) needs of the user population of the system on which the records are being loaded. For example, if the users of the system are chiefly monolingual, the removal of a single foreign character from a record may have little repercussion. On the contrary, if one considers the removal of all Chinese characters from a system serving an area where there are many readers of the language, the consequences may be considerable.
d) the need to record the original Unicode characters for future Unicode export or internal system conversion.

In this report, the recommended choice is Option 4 above.
The reasons for the choice of this option are:
a) the original Unicode value can be reconstructed
(no information is lost and the record can be reverted to Unicode).
b) the method is already established as a common method of dealing with this problem in the IT industry.
c) this option is not expensive or complex to program (Option 5 is somewhat more expensive and error prone although it is more user-friendly).

2.2 Moving Records between Systems that use Different Unicode Versions

There can be an issue of missing characters also within between Unicode systems which are running different versions of Unicode. For example, a system running Unicode version 4.0 sends a character code to a system running Unicode 3.2 and that character is not present in Unicode 3.2.

2.2.1 Recommendation

This report recommends using the same option as that used in moving records to a non-Unicode system using the current MARC8 repertoire—replace the character with a representation of the hexadecimal code of the character using ASCII representation.

2.3 Display and Fonts Issues

The move to full Unicode will immediately raise for libraries the issue of the availability of fonts and the characteristics of the fonts that are in fact available.

> "The difference between identifying a character and rendering it on screen or paper is crucial to understanding Unicode's role in text processing. The character identified by a Unicode code point is an abstract entity, such as "LATIN CAPITAL LETTER A" or "BENGALI DIGIT FIVE". The mark made on screen or paper, called a glyph, is a visual representation of the character.
>
> "The Unicode standard does not define glyph images. That is, the standard defines how characters are interpreted, not how glyphs are rendered. Ultimately, the software or hardware rendering engine of a computer is

> responsible for the appearance of the characters on the screen. The Unicode standard does not specify the precise shape, size, or orientation of on-screen characters." (The Unicode Consortium. *The Unicode standard, version 4.0*. Reading, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1, page 5)

There are no fonts currently in existence that will display all characters in Unicode version 4.0. A single TrueType or OpenType font, which has a limit of 65,536 characters, is no longer then capable of including all Unicode characters.

2.3.1 Large comprehensive fonts

Two fonts are available however which have relatively good coverage of characters in languages commonly encountered. These are: "Arial Unicode" from Microsoft and "Andale Mono" from Agfa-Monotype. (Both fonts have actually been built by Agfa Monotype but the former is available only from Microsoft—by license.) The Arial Unicode font is based on Unicode version 2.1. (Unicode 2.1 includes 38,871 graphic characters.) It does include a few additions beyond version 2.1 but there are no plans to expand it with enormous numbers of new characters. It is planned however that there will be "a new version that has better quality and more complete coverage of the scripts it already supports". These additions will expand this font in some areas to bring it up to Unicode 3.2 and in other areas up to Unicode 4.0.

Andale Mono supports all characters in Unicode 3.0—that is, 49,170 graphic characters. The major difference between Unicode versions 2.1 and 3.0—except for Canadian Aboriginal Syllabics—is the addition of "Han Extension A" with 6,582 additional han characters. Either of these fonts would cover the vast majority of needs of most research libraries. Andale Mono is also currently undergoing upgrades that will bring it up to Unicode 4.0; the problem of size will be addressed by putting some of the CJK additions in separate fonts.

In addition to simply adding more characters, Agfa Monotype will also be adding font capabilities which will support most dynamic diacritic plus letter combinations in the following scripts: Latin, Cyrillic and Greek.
Both of these large fonts are available commercially and are subject to licensing restrictions.

2.3.2 Linked sets of fonts

By policy, Microsoft and large font houses have decided that no more work will be done on comprehensive fonts and that all effort will be devoted instead to the creation of "linked sets of fonts". Partly, this is due to the 65K (65,536) limit on fonts that Unicode 4.0 now goes well beyond. Another factor relates to the requirements of font design—for example, a well-designed Thai font has different requirements from a well-designed font for Canadian Aboriginal Syllabics and font designers are unwilling to create comprehensive fonts that do not do justice to many of the characters that would be included. From the point of view of serving a market, this concept is especially true of fonts that include han characters. Adobe, for example, has created fonts specifically for

Japan or for China in which tens of thousands of glyph variants were created to make the font completely acceptable in each of those markets.

2.3.3 Situation of libraries regarding fonts

For libraries collecting little non-English library material there will not be significant font problems or issues with the move to full Unicode. The fonts which are regularly available should be completely adequate in the handling of English and in fact most European language material. Characters which are beyond the current MARC8 repertoire such as the section sign (§)(U+00A7), the Yen sign (¥) (U+00A5)—to give only two examples—will appear just as they do in this document since they are regular characters in the commonly available fonts such as "Times" and "Arial". And it is unlikely, for many libraries, that very many characters will be needed that are not available in this way.

However, when we consider the needs of large research libraries the situation is somewhat different. The move to full Unicode could readily involve characters or whole scripts that are not present in the fonts that are regularly available in the personal computers that usually are used as the interface equipment for most library automation systems. What then can be done?

Libraries can consider licensing one of the two comprehensive fonts listed above—various licensing arrangements are possible. Either of these comprehensive fonts would be capable of handling the vast majority of characters required by even a major research library.

Secondly, libraries can investigate the availability of a linked set of fonts covering the scripts in which the library specifically collects materials.

2.3.4 Terminal software, browsers, word processors

In considering requirements for display it is relevant to point out the universal trend away from custom software for display on terminals of a library system toward the use of commonly available browser software maintained at someone else's expense. Therefore, issues of display need to be studied in the context of the capabilities of the browsers available to the users of the library system in question. Display capabilities will certainly vary from one browser to another and even from one version of the same browser to another version of that browser. Recommendations for font use will need to be correlated with browser choice and capability. And it should also be made clear that capabilities experienced with a word processor may or may not be reflected in the browser of choice.

2.3.5 Combining letters and diacritics in display

(See also: Task 2: Multiple Encodings)

There can be issues with fonts in the combining of letters with the diacritics that go with them. According to one expert consulted in the course of this study, the library will "need fonts and applications that support accurate dynamic mark positioning for all the scripts you want to cover." In other words, the font in question will need to be able to recognize the presence of a diacritic and know how to display that diacritic appropriately positioned with the character in question. A statement from a Microsoft representative notes that:

"combining diacritic display (for transliterations and African languages that require them) was added to Word2003, and there are several fonts now that support the necessary OpenType features to allow this. Support will grow in the future".

In order to avoid combining difficulties for the font, some vendors will develop tables which convert separately stored character plus diacritic combinations to precomposed characters present in the fonts being delivered with the system. However, it is inevitable that certain precomposed combinations will not be present in the fonts supplied and in this case the font will have to perform the combination dynamically. If the font is not capable of making such a combination, the result may be that the character and the diacritic appear in separate character spaces. This is likely to be true for characters such as the Russian ligature diacritics that are placed in a compatibility area of Unicode and for this reason are not likely to be supported by the dynamic font combining capabilities of commercially available fonts. Although having the diacritic and letter appearing each in their own spaces is not good looking, it should be accepted for the time being—especially since the industry is moving clearly in the direction of smarter fonts and font engines which will support combining in the future.

Note that the following web site provides lists of fonts by Unicode code range:
http://www.alanwood.net/unicode/fontsbyrange.html
And it should also be noted that recent versions of Internet Explorer have, under "Tools/Internet Options/Fonts/" a dialogue box in which "Language scripts" can be specified. The corresponding fonts available on the PC concerned, that are associated with the scripts listed and have code points present in the range of that script, are searched for and displayed by the IE software.

Resources

1 MARC Web Site

The MARC21 Standards web site maintained by the Library of Congress (http://www.loc.gov/marc/) assembles much vital and useful information. In particular, the conversion tables between MARC8 encoding and Unicode appear at:
http://www.loc.gov/marc/specifications/specchartables.html
Also in print form:
USMARC Specifications for Record Structure, Character Sets, and Exchange Media. Washington, DC: Cataloging Distribution Service, Library of Congress.

2 Unicode Web Site

The Unicode web site (www.unicode.org) is also a key site relating to the topic of this study. The following special elements of it should be mentioned:

2.1 Unicode 4.0 Book and CD

Citation: The Unicode Consortium. *The Unicode standard, version 4.0*. Reading, MA, Addison-Wesley, 2003. ISBN 0-321-18578-1
Content:

2.2 Unicode Character Database

This database is available at: http://www.unicode.org/Public/UNIDATA or ftp://www.unicode.org/Public/UNIDATA.
Specific tables within this database are mentioned in the report above.

2.3 Unicode Cross Mapping Tables

These tables provide mappings between Unicode and various national standards. They are to be found at;
http://www.unicode.org/Public/MAPPINGS

2.4 Unicode Technical Documentation

Another source of extremely useful technical information are the technical documents which are posted at:
http://www.unicode.org/reports/index.html
There are a series of "Annexes" which are considered part of the standard itself: Those currently listed are:
UAX 9 The Bi-Directional Algorithm
UAX 11 East Asian Width
UAX 14 Line Breaking Properties
UAX 15 Unicode Normalization Forms
UAX 24 Script Names
UAX 29 Text Boundaries
Under Technical Standards, it is useful to consult UTS 10 Unicode Collation Algorithm when working on normalization forms.
Under Technical Reports the following are of particular relevance to this report:
UTR 17 Character Encoding Model
UTR 22 Character Mapping Tables
UTR 23 Character Properties
UTR 30 Character Foldings

3 W3C Documents and Software

The W3C organization has a site (http://www.w3c.org/International/charlint/) which provides a software tool, written in PERL, which will check for conformance to Normalization Form C of the Unicode Technical Report # 15.

Appendix A: Unicode Scripts

Scripts and counts are only for the BMP (Basic Multilingual Plane) of Unicode. A list of code points for each script will be found in the Unicode table "Scripts.txt" on the Unicode site (www.unicode.org).
"New" means that no characters of the script are in MARC8.
"Partial" means that some characters of the script are present in MARC8 but full Unicode has more characters than MARC8.

Script	New/Partial	MARC8 Count*	Unicode Count (4.0)
Arabic	Partial	56	877
Armenian	New	0	83
Bengali	New	0	81
Bopomofo	New	0	64
CAS**	New	0	628
Cherokee	New	0	85
Cyrillic	Partial	102	260
Devanagri	New	0	102
Ethiopic	New	0	337
Georgian	New	0	79
Greek	Partial	58	348
Gujarati	New	0	82
Gurmukhi	New	0	76
Hangul	Partial	2,028	11,558
Han	Partial	13,478***	27,814 (in BMP)
Hebrew	Partial	45	74
Hiragana	Partial	88	89
Kannada	New	0	81
Katakana	Partial	83	164
Khmer	New	0	94
Lao	New	0	65
Latin	Partial	74	938
Malayalam	New	0	78
Mongolian	New	0	140
Myanmar	New	0	72
Ogham	New	0	26
Oriya	New	0	80
Runic	New	0	78
Sinhala	New	0	79
Syriac	New	0	62
Tamil	New	0	61
Telugu	New	0	80
Thaana	New	0	50
Thai	New	0	83

Script	New/Partial	MARC8 Count*	Unicode Count (4.0)
Tibetan	New	0	139
Yi	New	0	1220

*MARC8 counts include characters but not punctuation, symbols or combining marks.
**Canadian Aboriginal Syllabics. Accepted for inclusion in MARC8 in Unicode, 2002.
***This count includes 10 characters missing from the published standard.

Appendix B: Unicode 4.0 Blocks and MARC8 Encoding

Note: this chart includes all Unicode 4.0 blocks including those beyond the Basic Multilingual Plane (Plane 0)
"Same" means that the MARC8 table and the Unicode table have essentially the same character repertoires.
"New" means that none of the characters in the Unicode block are in MARC8 repertoire.
"Partial" means that some of the characters in this Unicode block are in the MARC8 repertoire and some are not. For most of the blocks listed as "Partial" below tables have been developed showing which characters are new to MARC8 and which are already in MARC8.

Begin	End	Block Name	MARC8 Status
0000	007F	Basic Latin	Same
0080	00FF	Latin-1 Supplement	Partial 01
0100	017F	Latin Extended-A	Partial 02
0180	024F	Latin Extended-B	Partial 03
0250	02AF	IPA Extensions	Partial 04
02B0	02FF	Spacing Modifier Letters	Partial 05
0300	036F	Combining Diacritical Marks	Partial 06
0370	03FF	Greek and Coptic	Partial 07
0400	04FF	Cyrillic	Partial 08
0500	052F	Cyrillic Supplementary	Partial 08
0530	058F	Armenian	New
0590	05FF	Hebrew	Partial 09
0600	06FF	Arabic	Partial 10
0700	074F	Syriac	New
0780	07BF	Thaana	New
0900	097F	Devanagari	New
0980	09FF	Bengali	New
0A00	0A7F	Gurmukhi	New
0A80	0AFF	Gujarati	New
0B00	0B7F	Oriya	New
0B80	0BFF	Tamil	New
0C00	0C7F	Telugu	New
0C80	0CFF	Kannada	New
0D00	0D7F	Malayalam	New
0D80	0DFF	Sinhala	New
0E00	0E7F	Thai	New
0E80	0EFF	Lao	New
0F00	0FFF	Tibetan	New
1000	109F	Myanmar	New
10A0	10FF	Georgian	New

Begin	End	Block Name	MARC8 Status
1100	11FF	Hangul Jamo	New
1200	137F	Ethiopic	New
13A0	13FF	Cherokee	New
1400	167F	Unified Canadian Aboriginal Syllabics	New
1680	169F	Ogham	New
16A0	16FF	Runic	New
1700	171F	Tagalog	New
1720	173F	Hanunoo	New
1740	175F	Buhid	New
1760	177F	Tagbanwa	New
1780	17FF	Khmer	New
1800	18AF	Mongolian	New
1900	194F	Limbu	New
1950	197F	Tai Le	New
19E0	19FF	Khmer Symbols	New
1D00	1D7F	Phonetic Extensions	New
1E00	1EFF	Latin Extended Additional	New
1F00	1FFF	Greek Extended	New
2000	206F	General Punctuation	New
2070	209F	Superscripts and Subscripts	Partial 11
20A0	20CF	Currency Symbols	Partial
20D0	20FF	Combining Diacritical Marks for Symbols	New
2100	214F	Letterlike Symbols	New
2150	218F	Number Forms	New
2190	21FF	Arrows	New
2200	22FF	Mathematical Operators	New
2300	23FF	Miscellaneous Technical	New
2400	243F	Control Pictures	New
2440	245F	Optical Character Recognition	New
2460	24FF	Enclosed Alphanumerics	New
2500	257F	Box Drawing	New
2580	259F	Block Elements	New
25A0	25FF	Geometric Shapes	New
2600	26FF	Miscellaneous Symbols	New
2700	27BF	Dingbats	New
27C0	27EF	Miscellaneous Mathematical Symbols-A	New
27F0	27FF	Supplemental Arrows-A	New
2800	28FF	Braille Patterns	New
2900	297F	Supplemental Arrows-B	New
2980	29FF	Miscellaneous Mathematical Symbols-B	New
2A00	2AFF	Supplemental Mathematical Operators	New
2B00	2BFF	Miscellaneous Symbols and Arrows	New
2E80	2EFF	CJK Radicals Supplement	New
2F00	2FDF	Kangxi Radicals	New

Begin	End	Block Name	MARC8 Status
2FF0	2FFF	Ideographic Description Characters	New
3000	303F	CJK Symbols and Punctuation	Partial 12
3040	309F	Hiragana	Partial 13
30A0	30FF	Katakana	Partial 13
3100	312F	Bopomofo	New
3130	318F	Hangul Compatibility Jamo	New
3190	319F	Kanbun	New
31A0	31BF	Bopomofo Extended	New
31F0	31FF	Katakana Phonetic Extensions	New
3200	32FF	Enclosed CJK Letters and Months	New
3300	33FF	CJK Compatibility	New
3400	4DBF	CJK Unified Ideographs Extension A	
4DC0	4DFF	Yijing Hexagram Symbols	New
4E00	9FFF	CJK Unified Ideographs	Partial 14
A000	A48F	Yi Syllables	New
A490	A4CF	Yi Radicals	New
AC00	D7AF	Hangul Syllables	Partial 15
D800	DB7F	High Surrogates	New
DB80	DBFF	High Private Use Surrogates	New
DC00	DFFF	Low Surrogates	New
E000	F8FF	Private Use Area	
F900	FAFF	CJK Compatibility Ideographs	
FB00	FB4F	Alphabetic Presentation Forms	New
FB50	FDFF	Arabic Presentation Forms-A	New
FE00	FE0F	Variation Selectors	New
FE20	FE2F	Combining Half Marks	Same
FE30	FE4F	CJK Compatibility Forms	New
FE50	FE6F	Small Form Variants	New
FE70	FEFF	Arabic Presentation Forms-B	New
FF00	FFEF	Halfwidth and Fullwidth Forms	Partial
FFF0	FFFF	Specials	New
10000	1007F	Linear B Syllabary	New
10080	100FF	Linear B Ideograms	New
10100	1013F	Aegean Numbers	New
10300	1032F	Old Italic	New
10330	1034F	Gothic	New
10380	1039F	Ugaritic	New
10400	1044F	Deseret	New
10450	1047F	Shavian	New
10480	104AF	Osmanya	New
10800	1083F	Cypriot Syllabary	New
1D000	1D0FF	Byzantine Musical Symbols	New
1D100	1D1FF	Musical Symbols	New
1D300	1D35F	Tai Xuan Jing Symbols	New

Begin	End	Block Name	MARC8 Status
1D400	1D7FF	Mathematical Alphanumeric Symbols	New
20000	2A6DF	CJK Unified Ideographs Extension B	
2F800	2FA1F	CJK Compatibility Ideographs Supplement	
E0000	E007F	Tags	New
E0100	E01EF	Variation Selectors Supplement	New
F0000	FFFFF	Supplementary Private Use Area-A	New
100000	10FFFF	Supplementary Private Use Area-B	New