ＩＳＯ１０６４６と国際化された文字コードについて

太田　昌孝

東京工業大学総合情報処理センター

全世界の文字を含むコード体系としてＩＳＯで現在策定中の規格である１０６４６は、各国漢字の混在、方向性の異なる言語の混在などへの対応はおざなりである。そこで、国際化に必要な各種付加情報を含めたエンコーディングであるＩＣＯＤＥと、それを既存のバイトストリーム中で利用するための可変長エンコード技法であるＩＵＴＦを設計した。

1. はじめに

世界各国で使われている総ての文字に共通に使える文字コードを作ろうという動きの結果、ＩＳＯは、ＩＳＯ１０６４６ － Ｕｎｉｖｅｒｓａｌ Ｃｏｄｅｄ Ｃｈａｒａｃｔｅｒ Ｓｅｔ（ＵＣＳ）の制定作業を行っている。そのドラフト第一版のＤＩＳ１０６４６［1］は、もっぱらヨーロッパ系の国々からの、「同様の試みであるＵｎｉｃｏｄｅ［2］と統一されるべきだ」という理由により、採用が否決された。その統一方法が検討された結果生まれたドラフト第二版であるＤＩＳ１０６４６－１．２は、第一版の面影をほとんどとどめず、ヨーロッパ言語に限った利用には有効であるが、国際化文字コードとしてはごく単純な用途でも使い物にならないものとなった。しかし、日本の反対にもかかわらず採用が票決され、現在ＩＳ化にむけての事務手続き中である。

ＤＩＳ１０６４６－１．２の大きな欠点の一つは、Ｈａｎ Ｕｎｉｆｉｃａｔｉｏｎである。Ｈａｎとは漢字のことであるが、漢字を利用している主な国である中国（含台湾）、日本、韓国で使っている、共通の祖先をもつ漢字を同じものとみなして同じ文字コードを割り当ててしまおうというものである。これは、１６ビットの文字コードとして米国計算機メーカーを中心に開発された文字コードであるＵｎｉｃｏｄｅで採用された考えである。数万はある漢字を含めた文字を１６ビットで表すためには当然必要となる処置ではあるが、このような文字コードは、当然のことながら、アジア圏では使い物にならない。日本語と中国語では、対応する漢字とはいえかなり形が違うものも多いにもかかわらず、それらには同じ文字コードが割り振られている。例えば、「草」という字の草冠は、中国漢字では、「十」が二つ横に並んだ形になるが、同一視されている。また、たとえ日中で同じ形と見なせる文字も、印刷の場合は異なる字形で表現するのが日中の言語が入り交じった場合の通常の表記法である。漢字仮名混じり文を表記する日本漢字と、漢文のための中国漢字では、当然のごとく字体が違う。が、文字コードを統一されてしまっては、この区別もできない。

ISO10646 and Internationalized Coded Character Set
by Masataka Ohta (Tokyo Institute of Technology)

また、もっぱらインドでさまざまな言語の表記に利用されているデバナガリ文字は、複数の母音と子音が組みあわさって一つの文字を形作る。ところが、この組みあわせの規則は、言語により異なる。それにもかかわらず、ＤＩＳ１０６４６－１．２では個々の母音と子音にのみコードが振られているため、言語毎に組みあわせ方法を変更することは不可能である。

こういった欠陥はヨーロッパ系の言語を使う限りはまったく問題とならないが、アジア圏では、なんらかの言語を識別する情報を付加しないことには、まともな表示もできないわけである。

ＤＩＳ１０６４６－１．２自体は１６ビットまたは３１ビットの固定長のコードである。３２ビット目は常に０であり、ユーザー用のフラグビットとして利用できる。現在のところ下位１６ビット以外のビットは総て０であり、１６ビットコードであるＵｎｉｃｏｄｅと全く同じものとなっている。各国の文字の割り当て状況を図１に示す。図は、下位８ビットが左から右に横軸、上位８ビットが上から下に縦軸となっており、左端の１６進数が上位８ビットである。図でわかるとおり、ＡＳＣＩＩ文字（ＩＳＯ－６４６　ＩＲＶ）は、上位９ビットが０の位置に割り当てられており、７ビットのＡＳＣＩＩコードとまったく同じコードとなっている。図の、４Ｅ～９Ｆの部分が、問題のＨａｎ　Ｕｎｉｆｉｃａｔｉｏｎが行われている部分であるが、領域としては２万文字程度しか収納できず、５万を軽く越える漢字総てが表現できないのも深刻な問題である。もはや、Ｕｎｉｆｉｃａｔｉｏｎの理由づけであった、「総ての文字を１６ビットでエンコードする」という目的は、破綻しているわけである。

さて、ＤＩＳ１０６４６－１．２を既存のシステムの上で利用するため、ＡＳＣＩＩコードと共存可能の可変長コードＵＴＦが提案されている。ＤＩＳ１０６４６－１．２にあるＵＴＦを以後ＯＵＴＦ（＝Ｏｒｉｇｉｎａｌ　ＵＴＦ）と呼ぶ。

ＯＵＴＦは、利用に際して各種の不便があったため、Ｘ／Ｏｐｅｎが現在提案しているＵＴＦ（以後ＸＵＴＦと呼ぶ）では、後に述べる各種の「改良」が施されている。付録Ａのように、ＸＵＴＦでは、ＡＳＣＩＩ文字は１バイトでそのまま表現可能である。いっぽう、文字コード２０４７までの文字は、２バイトで表現され、それ以上の文字は３バイトとなる。図１をみてのとおり、２バイトで表現可能なのは、ヨーロッパ、中近東の言語であり、アジア系の言語は冷遇されている。もちろん、ひらがな、カタカナも３バイトとなる。

というわけで、世界各国で使える共通の文字コードを作ろうという動きは、日本人の目からみて、うまくいったとはいいがたい。とはいえ、捕鯨問題同様、欧米人に道理は通じなかった。

2. ＩＣＯＤＥ

今後は、ＸＵＴＦに従った製品が、外国メーカーから現実のものとして発売されることが予想される。

そこで、対応策の一つとして、国際化を考慮しＵｎｉｃｏｄｅに言語情報等を付加し、Ｕｎｉｃｏｄｅ上位互換なコードを設計した。このコードを、ＩＣＯＤＥ（＝Ｉｎｔｅｒｎａｔｉｏｎａｌｉｚｅｄ　ＣＯＤＥ）と名付ける。

同時に、Ｕｎｉｃｏｄｅと上位互換なエンコーディングとして、ＩＵＴＦ（＝Ｉｎｔｅｒｎａｔｉｏｎａｌｉｚｅｄ　ＵＴＦ）を設計した。

上位互換ということで、ＩＣＯＤＥは、下位１６ビットは、まったくＵｎｉｃｏｄｅと同じである必要がある。そこで、ＩＣＯＤＥの下位１６ビットは、Ｕｎｉｃｏｄｅをそのまま、すなわちＤＩＳ１０６４６－１．２の

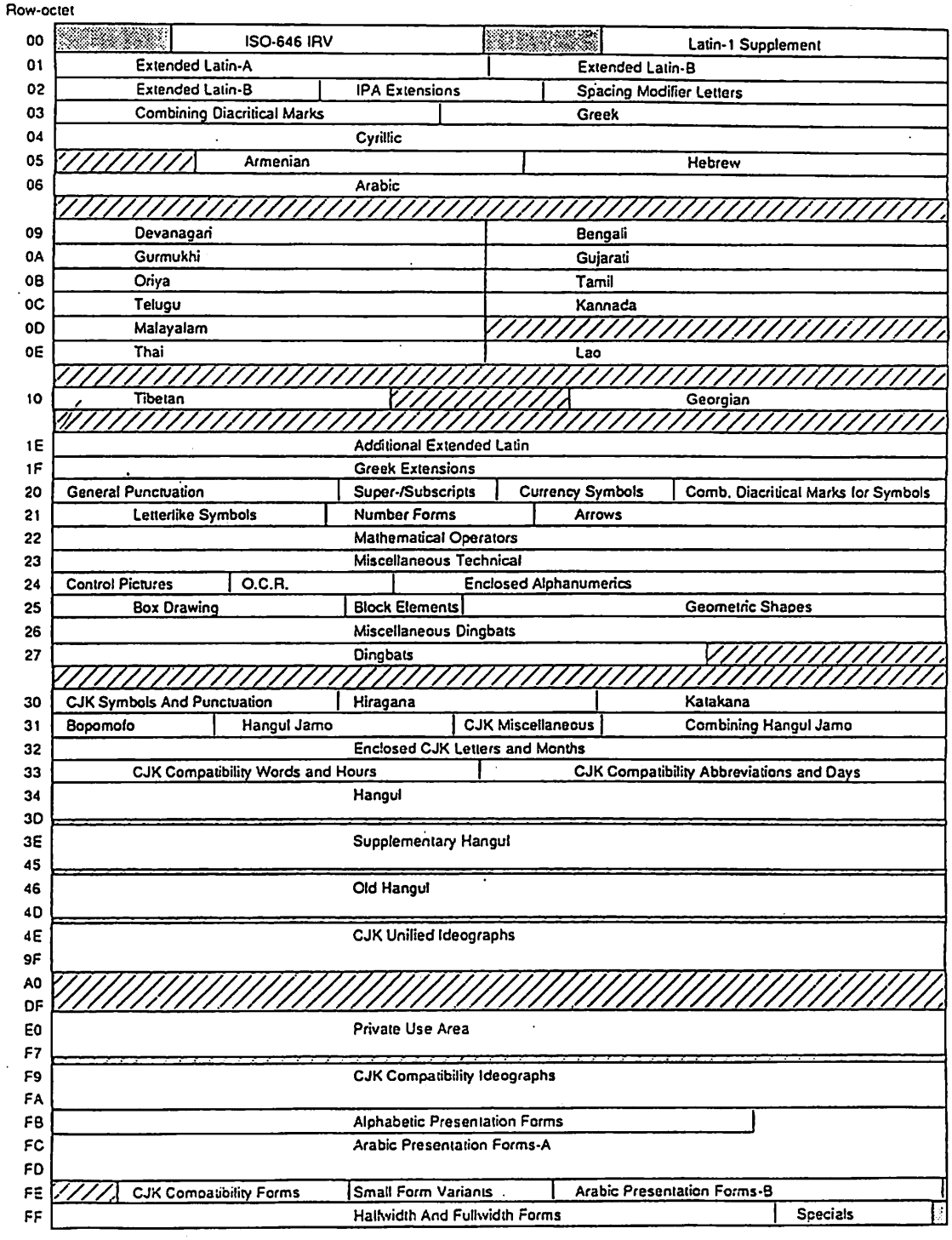

図1　ＤＩＳ１０６４６－１．２の文字の割り当て（ＤＩＳより引用）

下位１６ビットをそのまま使うこととする。また、Ｕｎｉｃｏｄｅでは総ての漢字は表現できないため、文字種の拡張のためのビットを１ビットその上に付加し、１７ビットのコードとする。どのみちＤＩＳ１０６４６－１．２は、１７ビット程度への拡張は必要であろう。が、これがＩＣＯＤＥの１７ビット目と同じになるかどうかは、ここでは規定しない。ＩＳＯの今後の動き次第である。

１節で述べたように、言語識別情報等は処理コード中に必要である。言語識別情報としては、とりあえず漢字には３ビット必要となる。漢字の場合、

0　ＡＮＹ

1　中国漢字

2　台湾漢字

3　日本漢字

4　韓国漢字

5　ベトナム漢字

のようにエンコードする。これを１７ビットコードに付加して、全体で２０ビットのコードを得る。

インド等には数百の言語があるといわれるが、文字種字体は漢字と比べて少ないので、その識別には下位１７ビットの部分が利用できる。

さて、アラビア語等の言語は、他の多くの言語と異なり、横書きであるがもっぱら右から左に表記する。このような性質を方向性というが、方向性の異なる言語を混在させて表記するためには、方向性識別情報を導入する必要がある。一般には方向性は入れ子構造をもつ。つまり、ある方向性の言語内に、異なる方向性を持つ言語が埋め込まれ、さらにその中に方向性が異なるものが埋め込まれることがある。このような入れ子構造は文章の意味的構造を反映したものであるため、文字コードレベルで単純に対処できる問題ではない。そこで、方向性が異なる文字を混在させるための最低限の処置として、本来の方向性とは逆転していることを示す方向性ビットを１ビット用意し：

0　本来の方向性

1　本来の方向性の逆

とする。そして、文字は表示の順に並べ、右から左に書きたい行内の文字は総て右から左の方向性、左から右に書きたい行内の文字は総て左から右の方向性となるように、方向性ビットで調節することとする。これだけの情報があれば、印刷などの最低限の用には足りる。

この結果全体で２１ビットのコードが得られる（図２）。これは、ＩＵＴＦで４バイトで表現できる。

場合によっては、さらにこの上に、ＤＩＳ１０６４６－１．２の上位１４（１５）ビットや、言語識別情報の上位ビットを付加し、ユーザ用フラグビット等の領域も考えて処理コードとすることになるが、これは今後の検討課題とし、当面は上位ビットは０とする。ただし、３２ビット目にはＤＩＳ１０６４６－１．２の３２ビット目である、ユーザ用のフラグビットをそのまま置く。

結果のコードは、ＵＣＳ４が３１ビット全部を利用すると全体として３２ビットを越える可能性もあるが、ＵＣＳ２にすら空きがある現状からみて、実際には越えることはないであろう。万一越えても、既に６４ビットプロセッサが実用化している現状では、深刻な問題とはならない。

3. ＩＵＴＦ

ＩＵＴＦでは、２バイトで表現可能なコード数を極力増やし、ここに、言語識別情報を含めてエンコードされたよく使われる漢字等を置く。エンコード方針は４節で述べる。

ＸＵＴＦの設計目標は、

1）ＵＮＩＸファイルシステムとの整合性

2）既存プログラムとの整合性

3）処理コードとの間の変換の容易さ

4）第一バイトでコード長が判明すること

5）バイト数が長くなりすぎないこと

6）ランダムアクセス後、文字区切りを効率良く発見できること

である。1）は、ＵＮＩＸファイルシステムとの整合性を考慮し、ＡＳＣＩＩ文字スラッシュのコードは、スラッシュに対してのみしか現れないようにするということである。

ＩＵＴＦの設計にあたっては、エンコード結果の同一性だけでなくこれらの目標にも配慮し、ＸＵＴＦ向けに書かれたアプリケーションプログラムが、ライブラリの入れ替えだけでそのままＩＵＴＦでも動作可能となるようにした。

ＩＵＴＦでは、1）、5）、6）の性質は、そのまま保存されている。

2）の意味するところは、アスキー以外の文字のエンコーディングには、ＭＳＢの立ったものを使うということである。しかし、ＵＴＦは処理コードではないので、この要求はＸＵＴＦ向けに書かれたプログラムに対しては無意味である。ＩＵＴＦはこの性質はもたない。

3）の性質は、変換のためのライブラリプログラムの複雑さの点では犠牲になったが、速度的に遜色のないようにできる。ＸＵＴＦにも含まれた表現の場合、処理速度はほぼ同等であるし、日本語等では、入出力量が少ないぶんかえって速くなるであろう。

4）の性質は、バイトの先読みを必要としないためには重要であるが、

4．1）第一バイトによりコード長が1バイトであるか2バイト以上であるか判明すること

4．2）第二バイトが存在する場合、そのバイトにより、コード長が判明すること

のように弱めても、やはり先読みは必要とならないので、そのようにした（3バイト目まで見る必要がある）。

ＩＵＴＦでは、もっぱら2）の性質を犠牲にすることにより、ＸＵＴＦに比べてより多くのエンコード領域が利用可能となった（詳細は付録Ｂ）。この領域は、ＩＺＯＮＥ（Ｉｎｔｅｒｎａｔｉｏｎａｌｉｚａｉｔｏｎ Ｚｏｎｅ）と呼び、短縮したエンコードと、国際化されたアプリケーションで必要な、言語識別情報等を含めたエンコードを行う。

ＸＵＴＦに現在表れる表現は、総てＩＵＴＦでも同じ意味を持つ。すなわち、０ｘｆｆｆｆ以下のＩＣＯＤＥは、ＩＵＴＦで読んでもＸＵＴＦと同じコードとして読まれる。

4．ＩＺＯＮＥ

ＩＺＯＮＥには、８２５６個の2バイト表現（以後、ＩＺＯＮＥ２と呼ぶ）と、３５Ｋ個ほどの3バイト表現（以後、ＩＺＯＮＥ３と呼ぶ）が存在し、効率的なエンコーディングに利用できる。

このＩＺＯＮＥへの実際の文字の割り当ては今後の検討課題であるが、基本的な方針は以下のようなものであるべきである。

ＩＺＯＮＥ２は、極めて限られた大きさしかないので、国際協調の元に使う必要がある。日本語用には、２９６５文字もあるＪＩＳＸ０２０８の第一水準漢字を全部入れることはあきらめ、かたかな、ひらがな、教育漢字、ごく少数の記号のみを入れよう。

頻出する中国漢字や、頻出するハングル（ハングルは既にＸＵＴＦにおいて、3バイトにエンコードされている）も、それぞれ数百～２０００個程度エンコードされるべきである。

頻出するタイ文字やデバナガリ文字も、ハングル同様にｐｒｅｃｏｍｂｉｎｅｄ ｃｈａｒａｃｔｅｒとして、数百～２０００程度エンコードされるべきである。

ＩＺＯＮＥ３には、その他のｐｒｅｃｏｍｂｉｎｅｄ　ｃｈａｒａｃｔｅｒ等をエンコードする。

４バイト以上の領域は、ＩＣＯＤＥをそのままエンコードすることに利用する。

5. 最後に

ＤＩＳ１０６４６－１．２は、これまで挙げた以外にも種々問題をかかえているが、それらの問題にもなんとか対処できると思われる。

謝辞

本稿を用意するにあたって、筆者が所属する「ＩＳＯ　１０６４６の利用技術に関する研究会」の皆様に様々な有益な御助言をいただきました。また、（株）シグマシステムには、研究会の運営のお世話になっています。ここに感謝の意を表明いたします。

付録Ａ．ＸＵＴＦの実際

１バイトで表現される値を次のように分類する。

```
C0:0~32,127
A :33~126
Tx:128~191
T1:192~223
T2:224~239
T3:240~247
T4:248~251
T5:252~253
Ty:254~255(未使用)
```

この時、ＸＵＴＦの文字のエンコードとして、

バイト列	文字コード
C0	0~32,127
A	33~126
T1 Tx	128~2047
T2 Tx Tx	2048~2^16-1
T3 Tx Tx Tx	2^16~2^21-1
T4 Tx Tx Tx Tx	2^21~2^26-1
T5 Tx Tx Tx Tx Tx	2^26~2^31-1

という組合せのみを許せば、バイト列中の任意の地点から、バイト列をせいぜい６バイト逆にたどることにより、文字の境界を確定することができる。

Ｕｎｉｃｏｄｅのエンコードには、

```
T2 Tx Tx
```

までの３バイトで十分である。

付録Ｂ．ＩＵＴＦの実際

１バイトで表現される値を次のように分類する。

```
C0:0~32,127
A :33~126
A':33~46,48~126
C1:128~159
Tx:128~191(C1を含む)
T1:192~223
T2:224~239(=S2+S3+S4+S6+S7)
S2:224~229
S3:230~235
S4:236~237
S6:238
S7:239
U1:240~255
```

この時、ＩＵＴＦの文字のエンコードとして、

バイト列	文字コード
C0	0~32,127
A	33~126
T1 Tx	128~4095
T2 Tx Tx	4096~65535

まではＸＵＴＦのものをそのまま利用し、さらに以下の表現を許す。

バイト列	文字数
C1	32
T1 A'	2976
T2 A'	1488
U1 A'	1488
U1 Tx	1024
T1 T2	512
T1 U1	512
U1 T2	256
S2 Tx A'	35712
S3 Tx A' Tx	>2~21
S4 Tx A' Tx Tx	>2~25
S6 Tx A' Tx Tx Tx Tx	>2~36
S7 Tx A' Tx Tx Tx Tx Tx	>2~42

その結果、ＡＳＣＩＩのＣ１制御文字がそのまま利用できる他に、新たに８２５６個の２バイト表現と３５７１２個の３バイト表現がえられることになる。

この時、ＸＵＴＦと同様、バイト列中の任意の地点からバイト列をせいぜい８バイト逆にたどることにより、文字の境界を確定することができる。

参考文献：

[1] DIS 10646 Universal Multiple-Octet Coded Character Set (UCS), ISO/IEC JTC1 SC2, 1990

[2] DIS 10646-1.2 Universal Multiple-Octet Coded Character Set (UCS), ISO/IEC JTC1 SC2, Oct. 1991

[3] The Unicode Standard, Version 1.0, Vol. 1, The Unicode Consortium, Addison-Wesley, Oct. 1991

ビット位置	＞＝２１	２０	１９～１７	１６	１５～０
内容	０	方向性	言語識別情報	拡張	Ｕｎｉｃｏｄｅ

図２　ＩＣＯＤＥのビット割り当て