文字コードとその実装

荒川靖弘

第 1 版（公開版） 2001 年 11 月 3 日

目次

1　はじめに

異なるプラットフォームでテキスト情報をやり取りする際に必ず問題になるのが「文字コード」の変換です。また近年はインターネットに代表される情報交換の「国際化」も意識されはじめ，それを受けて登場した Unicode との互換性も問題になっています。本テキストでは「文字コード」の問題について考えてみたいと思います。

2　文字コード

通常私達が「文字コード」と呼んでいるものは，実際には以下に示す 2 つの内容を含んでいます。

- 符号化文字集合（coded charactor set）
- 文字エンコーディング（charactor encoding）

「符号化文字集合」とは，その名の通り符号化（数値化）した文字の集合を示します。「文字エンコーディング」とは符号化文字集合を情報処理用の「データ」に変換する方式を示します。符号化されている文字情報を更にエンコーディング（符号化）するというのは一見奇妙に見えます。どうしてそのようになっているのか見ていくことにしましょう。

2.1　ISO/IEC 646 と IRV（US-ASCII）

表 1 はインターネットの世界では俗に「US-ASCII」と呼ばれている文字コードです。

ASCII（American National Standard Code for Information Interchange）はもともと ANSI（American National Standards Institute）[*1] 規格（ANSI X3.4）として決められていました。

一方，ANSI とは独立に ISO（International Organization for Standardization: 国際標準化機構）[*2] でも各国用の文字コード規格「ISO/IEC 646」が作られました。現在の ASCII（US-ASCII）は ISO/IEC 646 の IRV（International Reference Version: 国際基準版）と同一となっています。ISO/IEC 646 は 7 bit のコード空間を定義していて，大まかに以下の 4 つの領域で構成されています。（模式的に書くと表 2 のようになります）

1. 制御文字（control charactors）：00H～1FH
2. SP（SPACE）：20H
3. 図形文字（graphic characters）：21H～7EH
4. DEL（DELETE）：7FH

制御文字領域は「CL」とも呼ばれています。SP は空白文字で図形文字の一種と考えることもできます。図形文字（いわゆる「文字」）領域は「GL」とも呼ばれています。DEL はもともと紙テープ時代に削除用のコードとして使われていた時の名残のようです。意味的には制御文字の一種と考えられなくもないですが，制御文字としては定義されていません。SP と DEL は GL に含める場合があります。

ISO/IEC 646 は本来，符号化文字集合を定めたものですが，実際にはそのコードをそのまま上記のようにエ

[*1] http://www.ansi.org/ 参照。
[*2] http://www.iso.org/ 参照。

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

表 1　IRV/US-ASCII

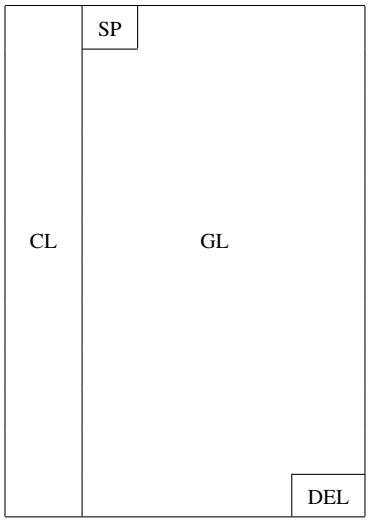

表 2　ISO/IEC 646

ンコーディングして使っています。

2.1.1　JIS ラテン文字

日本では ISO/IEC 646 に相当するものとして JIS X 0201 ラテン文字が定義されています。（JIS X 0201 については 2.2.1 章で改めて解説します）IRV と JIS ラテン文字との違いは以下のとおりです。

コード	IRV	JIS
5CH	\（REVERSE SOLIDUS）	¥（YEN SIGN）
7EH	~（TILDE）	‾（OVER LINE）

このように日本では IRV と JIS ラテン文字の違いは僅かです。しかし欧州などでは IRV との違いが顕著で各国間で全く互換が取れない状態になってしまいました。その結果（日本を除く）殆どの国は ISO/IEC 646 の各国語バージョンを使わなくなり，2.2 章で述べる ISO/IEC 8859 や「コードページ」にシフトしていきました。

2.2　ISO/IEC 8859 と JIS X 0201

各国間の文字コードの不整合を緩和するため，ISO では ISO/IEC 646 のコード空間を 8 bit に拡張した規格「ISO/IEC 8859」が作られました。ISO/IEC 8859 のレイアウトは ISO/IEC 646 を 2 つ並べたような形になっています。（模式的に書くと表 3 のようになります）

1. CL 制御文字（CL control charactors）：00H～1FH
2. GL 図形文字（GL graphic characters）：20H～7FH（含む SP,DEL）または 21H～7EH
3. CR 制御文字（CR control charactors）：80H～9FH
4. GR 図形文字（GR graphic characters）：A0H～FFH または A1H～FEH

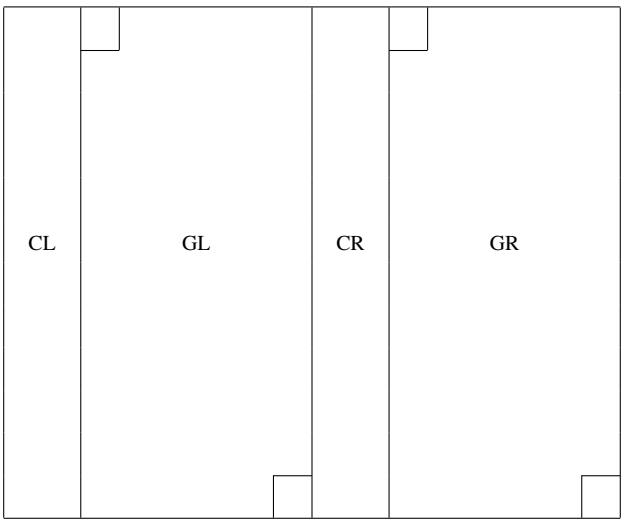

表 3　ISO/IEC 8859

通常 ISO/IEC 8859 の CL,GL 領域には IRV がそのまま収容されます。GR 領域には各国特有の文字が割り当てられました。ISO/IEC 8859 の実装で最も有名なのは ISO/IEC 8859-1（通称「Latin-1」）で，フランス語やドイツ語などの西欧でよく使われる文字が収容されています[*3]。

各国間の文字コードの互換性に悩んでいた（特に）欧州では，ISO/IEC 8859 は広く受け入れられ，現在もよく使われてます。

2.2.1 JIS X 0201

日本で ISO/IEC 8859 に相当するものとして JIS X 0201（当時は JIS C 6220 と呼ばれていました）があります。JIS X 0201 ではラテン文字と片仮名を定義していて，GL に（IRV ではなく）ラテン文字，GR に片仮名を配置しています。（表 4 参照）

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				0	@	P	`	p				ー	タ	ミ		
1			!	1	A	Q	a	q			。	ア	チ	ム		
2			"	2	B	R	b	r			「	イ	ツ	メ		
3			#	3	C	S	c	s			」	ウ	テ	モ		
4			$	4	D	T	d	t			、	エ	ト	ヤ		
5			%	5	E	U	e	u			・	オ	ナ	ユ		
6			&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
7			'	7	G	W	g	w			ァ	キ	ヌ	ラ		
8			(	8	H	X	h	x			ィ	ク	ネ	リ		
9			)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
A			*	:	J	Z	j	z			ェ	コ	ハ	レ		
B			+	;	K	[	k	{			ォ	サ	ヒ	ロ		
C			,	<	L	¥	l	|			ャ	シ	フ	ワ		
D			-	=	M	]	m	}			ュ	ス	ヘ	ン		
E			.	>	N	^	n	~			ョ	セ	ホ	゛		
F			/	?	O	_	o				ッ	ソ	マ	゜		

表 4　JIS X 0201（8 bit）

実は JIS X 0201 には 7 bit のエンコーディング方法も定義されています。（これについては 2.3.2 章で解説します）しかし，実際にはこの 8 bit エンコーディングの方が広く受け入れられました。

2.2.2 コードページ

一方，IBM/PC などでは「コードページ」という独自の体系が作られました。これは（CL,CR の制御文字領域も含めた）8 bit 空間をフルに使って図形文字を割り当てたものです。GL 領域はほぼ IRV（US-ASCII）ですがそれ以外は各国によって独自の図形文字が割り当てられています。そして国別情報を表す情報として「コー

[*3] ISO/IEC 8859 は 2001 年 3 月現在，ISO/IEC 8859-1～10,13～15 が発行済になっています。

ドページ」と呼ばれる値を PC 内に保持しています。日本語のページコードは「932」でほぼ JIS X 0201 に準拠した符号化がされています。現在の DOS/Windows システムにもコードページは残っていて，主に国別情報として使われています。

2.3　ISO/IEC 2022 によるエンコーディング

更に ISO では，符号化文字集合を動的に切り替え，複数バイト符号にも対応できるエンコーディング規格を作りました。これが ISO/IEC 2022 です。（ISO/IEC 2022 は日本では JIS X 0202 で定義されています）図 1 は ISO/IEC 2022 のエンコーディングの仕組みを模式的に表したものです。

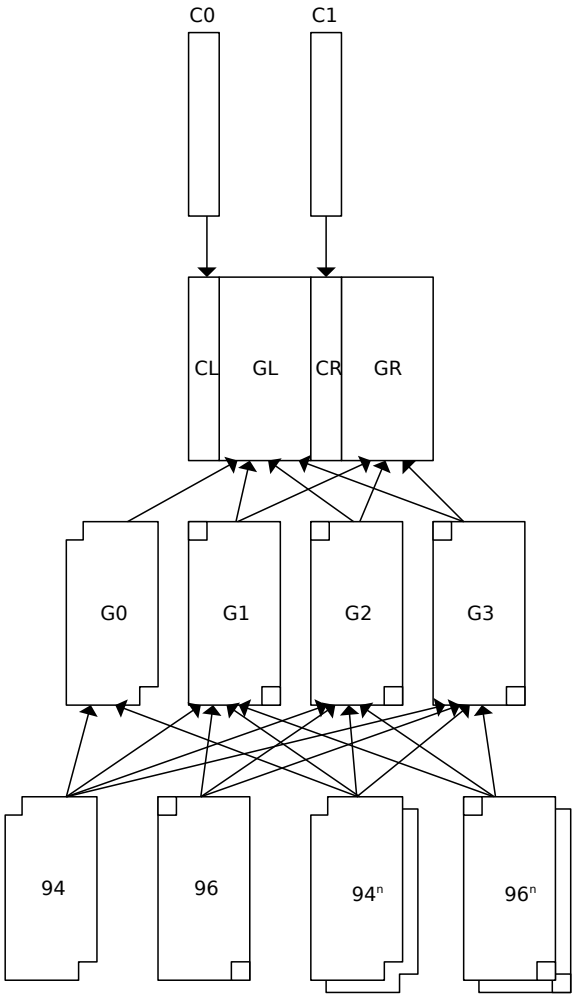

図 1　ISO/IEC 2022

これまでに登場した CL/CR/GL/GR で仕切られたコード空間を ISO/IEC 2022 では「コードテーブル」（JIS X 0202 では「符号表」）と呼びます。符号化された図形文字集合はコードテーブルに直接割り付けられるのではなく，G0〜G3 の「中間バッファ」に一旦「指示」され，中間バッファからコードテーブルの GL または GR に「呼び出」されます。図形文字集合には従来の 94 または 96 文字集合の他，複数バイト（94^n または 96^n）

文字の集合にも対応しています。ISO/IEC 2022 は CR/GR を使わないことで 7 bit エンコーディングにも対応できます。中間バッファからコードテーブルへの「呼び出し」は「ロッキングシフト」と呼ばれ C0/C1 制御文字（2.3.5 章参照）を使って行われます。（ただし，LS0/LS1 以外はエスケープシーケンスを用います）表 5 にそのシーケンスを示します。なお，G0 から GR への呼び出しはできないことになっています。

ロッキングシフト	シーケンス	
SI/LS0（G0 → GL）	0FH	[SI]
SO/LS1（G1 → GL）	0EH	[SO]
LS2（G2 → GL）	1BH 6EH	[ESC] n
LS3（G3 → GL）	1BH 6FH	[ESC] o
LS1R（G1 → GR）	1BH 7EH	[ESC] ~
LS2R（G2 → GR）	1BH 7DH	[ESC] }
LS3R（G3 → GR）	1BH 7CH	[ESC] |

シングルシフト	シーケンス	
SS2（G2 → GL/GR）	8EH または 1BH 4EH	[SS2] または [ESC] N
SS3（G3 → GL/GR）	8FH または 1BH 4FH	[SS3] または [ESC] O

表 5　ISO/IEC 2022 「呼び出し」シーケンス

シングルシフトは呼び出しの特別な形で，一時的に 1 文字だけ呼び出す場合に使用します。シングルシフトは G2 または G3 のみ使用でき，通常は GL へ呼び出しますが GR への呼び出しも可能です。

文字集合を G0〜G3 へ「指示」する場合もエスケープシーケンスを用います。表 6 に文字集合ごとの指示シーケンスを挙げます。96 文字集合または 96^n 文字集合を G0 に指示することはできません。94^n 文字集合から G0 への指示が 2 種類あるのは ISO/IEC 2022 の古い規格の名残です。*Ft* は符号化文字集合ごとに決められています。主なものを表 7 に挙げます。

このように ISO/IEC 2022 は非常に柔軟性の高い（逆に言えば複雑な）規格なのですが，この機能の全てを実装するのはあまり現実的ではありません。そこで実際には ISO/IEC 2022 の一部を省略した「サブセット」が使われます。

日本では JIS X 0201,0208,0212,0213 各規格で ISO/IEC 2022 に対応する文字エンコーディングを定義しています。JIS 0201 では，これまでも説明しましたが，ラテン文字と片仮名について定義しています。JIS X 0208,0213 は漢字集合について定義したもので，実装水準によって第一から第四まで決められています。JIS X 0212 は「補助漢字」と呼ばれ，もともと JIS X 0208 で不足している文字を補うために定義されたものですが，あまり使われることはなく，その後の JIS X 0213 の登場によって事実上 obsolete（時代遅れ，破棄された）な規格となってしまいました。

2.3.1　実装例: ISO/IEC 8859

ISO/IEC 8859 は ISO/IEC 2022 に準拠したエンコーディングであると見なすことができます。以下に簡単に概要を示します。

- G0 に IRV を指示

指示	シーケンス	
94 文字集合 → G0	1BH 28H *Ft*	`[ESC] ( Ft`
94 文字集合 → G1	1BH 29H *Ft*	`[ESC] ) Ft`
94 文字集合 → G2	1BH 2AH *Ft*	`[ESC] * Ft`
94 文字集合 → G3	1BH 2BH *Ft*	`[ESC] + Ft`
96 文字集合 → G1	1BH 2DH *Ft*	`[ESC] - Ft`
96 文字集合 → G2	1BH 2EH *Ft*	`[ESC] . Ft`
96 文字集合 → G3	1BH 2FH *Ft*	`[ESC] / Ft`
94^n 文字集合 → G0	1BH 24H 28H *Ft* または 1BH 24H *Ft*	`[ESC] $ ( Ft` または `[ESC] $ Ft`
94^n 文字集合 → G1	1BH 24H 29H *Ft*	`[ESC] $ ) Ft`
94^n 文字集合 → G2	1BH 24H 2AH *Ft*	`[ESC] $ * Ft`
94^n 文字集合 → G3	1BH 24H 2BH *Ft*	`[ESC] $ + Ft`
96^n 文字集合 → G1	1BH 24H 2DH *Ft*	`[ESC] $ - Ft`
96^n 文字集合 → G2	1BH 24H 2EH *Ft*	`[ESC] $ . Ft`
96^n 文字集合 → G3	1BH 24H 2FH *Ft*	`[ESC] $ / Ft`

表 6　ISO/IEC 2022「指示」シーケンス

種別	*Ft*	符号化文字集合名
94 文字集合	`@`	ISO/IEC 646 英国版
	`B`	ISO/IEC 646 IRV（US-ASCII）
	`H`	スウェーデン名前文字
	`I`	JIS X 0201 片仮名
	`J`	JIS X 0201 ラテン文字
96 文字集合	`A`	ISO/IEC 9959-1（GR 部分）
	`B`	ISO/IEC 9959-2（GR 部分）
	...	...
94^n 文字集合	`@`	JIS X 0208-1978（JIS C 6226）
	`A`	GB 2312-80（中国語簡体字）
	`B`	JIS X 0208-1983,1990,1997
	`C`	KS C 5601-1987（韓国語ハングル・漢字）
	`D`	JIS X 0212-1990
	`O`	JIS X 0213 1 面
	`P`	JIS X 0213 2 面

表 7　ISO/IEC 2022 文字集合

- G1 に ISO/IEC 8859-x GR 側を指示
- G0 を GL に呼び出す
- G1 を GR に呼び出す
- 指示・呼び出し用のシーケンスは使わない

2.3.2 実装例: JIS X 0201 7 bit エンコーディング

JIS X 0201 の 8 bit コーディングについては既に 2.2.1 章で説明しました。JIS X 0201 の 7 bit コーディングについても ISO/IEC 2022 に準拠したコーディングがあります。以下に概要を示します。

- G0 に JIS X 0201 ラテン文字（または IRV）を指示
- G1 に JIS X 0201 片仮名を指示
- G0 を GL に呼び出す
- JIS X 0201 片仮名は G1 を GL に呼び出すか直接 G0 に指示して使用
- シングルシフトは使わない

この方法なら G0 に他の符号化文字集合（JIS X 0208,0212,0213）を指示することでいくらでも拡張できます。後述する ISO-2022-JP（2.3.3 章）ではこのエンコーディング方法を拡張する形で実装されています。（ただし厳密には ISO/IEC 2022 に準拠していません）

2.3.3 実装例: ISO-2022-JP

ISO-2022-JP は RFC1468[*4] で定められるメッセージ交換用の規格で，JIS X 0208 の付属書でも言及されています。ISO-2022-JP は ISO/IEC 2022 を参考に設計されていますが，厳密に準拠していません。これは主に RFC2822[*5]（かつての RFC822）に配慮したものです。以下に概要を示します。

- 7 bit エンコーディング固定
- G0 に IRV（US-ASCII）を指示
- G0 を GL に呼び出す
- 文字集合の切り替えは G0 への指示で行い，以下の符号化文字集合の組み合わせのみ許す（JIS X 0201 片仮名は許されていない）

1BH 28H 42H	`[ESC] ( B`	ISO/IEC 646 IRV（US-ASCII）
1BH 28H 4AH	`[ESC] ( J`	JIS X 0201 ラテン文字
1BH 24H 40H	`[ESC] $ @`	JIS C 6226-1978
1BH 24H 42H	`[ESC] $ B`	JIS X 0208-1983,1990,1997

- ロッキングシフト・シングルシフトは使わない
- 改行コードを 0DH 0AH（`[CR] [LF]`）とする。
- テキストは IRV で始まる
- 行末（改行コードの手前）は必ず IRV または JIS X 0201 ラテン文字になっていること
- テキストは必ず IRV で終わらなければならない

[*4] http://www.asahi-net.or.jp/~bd9y-ktu/dtd_f/rfc_f/rfc1468j.html 参照。
[*5] Internet Message Format: http://www.puni.net/~mimori/rfc/rfc2822.txt 参照。

更に RFC1554[*6] では ISO-2022-JP を拡張した「ISO-2022-JP-2」を定義していて，JIS 補助漢字（JIS X 0212）や中国語簡体字，韓国語を指示することができます。また ISO/IEC 8859 の文字セットを G2 に指示し，シングルシフトで呼び出すことができるようにもなっています。おそらくこれは Unicode（2.5 章参照）を意識したエンコーディングと考えられなくもないですが，実際には殆ど見かけません。

ところで，JIS X 0213 では「ISO-2022-JP-3[*7]」が定義されています。ISO-2022-JP-3 では以下の符号化文字集合を指示できます。

1BH 28H 42H	`[ESC] ( B`	ISO/IEC 646 IRV（US-ASCII）
1BH 24H 42H	`[ESC] $ B`	JIS X 0213 1 面（ISO-2022-JP 互換用）
1BH 24H 28H 4FH	`[ESC] $ ( O`	JIS X 0213 1 面
1BH 24H 28H 50H	`[ESC] $ ( P`	JIS X 0213 2 面

JIS X 0213 の 1 面[*8] は従来の JIS 漢字第一・第二水準（JIS X 0208）と今回新しく追加された JIS 漢字第三水準までを含んだ集合で，2 面は JIS 漢字第四水準で追加された文字集合です。更に現在広く使われている ISO-2022-JP との互換性を保つために「`[ESC] $ B`」の指示シーケンスを 1 面として認めています。ただしこの場合には一部の文字が使えません。

ISO-2022-JP-3 はまだ RFC 化されていないようですが，一部のプラットフォームでは既に使われはじめています。

2.3.4 実装例: EUC-JP

EUC（Extended UNIX Code）は，文字通り UNIX 系プラットフォームにおいて多言語化の一環として開発されました。EUC の日本語サブセットのことを日本語 EUC または EUC-JP と呼びます。EUC も ISO/IEC 2022 に基づいて設計されています。EUC-JP を例にとってみます。

- G0 に IRV（または JIS X 0201 ラテン文字）を指示
- G1 に JIS X 0208 を指示
- G2 に JIS X 0201 片仮名を指示
- G3 に JIS X 0212 を指示
- G0 を GL に，G1 を GR に呼び出す
- G2 と G3 はシングルシフトで使用
- ロッキングシフトは使わない

指示が固定で呼び出しのためのロッキングシフトを使わないので，非常にすっきりしたコード体系になっているのが特徴です。ただし，EUC-JP では JIS X 0201 片仮名が冷遇されていて，シングルシフトを含めた 2 byte コードになっています。実際，つい最近まで JIS X 0201 片仮名や JIS X 0212 の補助漢字を実装していない UNIX 系プラットフォームが多く存在していたそうで，それが ISO-2022-JP において JIS X 0201 片仮名が排除された原因のひとつだとも言われています。

[*6] http://www.asahi-net.or.jp/~bd9y-ktu/dtd_f/rfc_f/rfc1554j.html 参照。

[*7] http://www.asahi-net.or.jp/~wq6k-yn/code/enc-x0213.html 参照。

[*8] JIS X 0213 は JIS X 0208 と共に使うことが前提となっています。JIS X 0208 の符号化文字集合は 94 区 ×94 点の 2 次元構成でしたが，JIS X 0213 では 2 面 ×94 区 ×94 点の 3 次元構成になっています。

EUC-JP にも JIS X 0213 において対応するエンコーディング（EUC-JISX0213）が定義されています。この場合は G1 に JIS X 0213 の 1 面を，G3 に JIS X 0213 の 2 面を指示します。

2.3.5 制御文字集合

ところで，今まで殆ど触れませんでしたが，制御文字集合は ISO/IEC 6429（日本では JIS X 0211）で定義されています。よく使われているものを表 8 に挙げます。

形式	制御文字		
C0	00H	NUL	NULL（空）
	08H	BS	BACKSPACE（後退）
	09H	HT	CHARACTER TABULATION（文字タブ）
	0AH	LF	LINE FEED（改行）
	0CH	FF	FORM FEED（書式送り）
	0DH	CR	CARRIAGE RETURN（復帰）
	0EH	SO/LS1	SHIFT-OUT（シフトアウト）/LOCKING-SHIFT ONE（ロッキングシフト 1）
	0FH	SI/LS0	SHIFT-IN（シフトイン）/LOCKING-SHIFT ZERO（ロッキングシフト 0）
	1BH	ESC	ESCAPE（エスケープ）
C1	8EH	SS2	SINGLE-SHIFT TWO（シングルシフト 2）
	8FH	SS3	SINGLE-SHIFT THREE（シングルシフト 3）

表 8　主な制御文字

このうち ESC は特別な制御文字で，C0 が呼び出されない状態でも有効になっています。BS は合成文字（「â」など）を表現する際に使われるのですが，プラットフォームによって対応していないことが多く，あまり使われません。（ISO-2022-JP などでは使ってはいけないことになっています）

RFC2822 でも定義されている制御文字があります。NUL は許されないコードとして定義されています。CR/LF が連続して現れる場合は改行コードと見なされます。更に HT および FF は MIME[*9] において「事実上の標準」となっています。

2.4 シフト JIS

シフト JIS は DOS/Windows や Macintosh などで使われるコード体系で，商業的に大きなシェアを占めています。シフト JIS の起源は日本初の 16 bit パソコンである三菱の「Multi16」に搭載された CP/M-86 の日本語対応版に実装されたものであると言われています。処理の容易なコードとして Microsoft など数社が策定したもので，「MS 漢字コード」とも呼ばれています。

シフト JIS は ISO/IEC 2022 非準拠です。簡単にいうと JIS X 0201 8 bit 空間（表 4 参照）の空いてる領域（CR の全てと GR の使われていない領域）に JIS X 0208 を無理矢理詰め込んだような構成になっています。しかも，漢字コードの 2 byte 目は GL の一部を含んでいます。

シフト JIS は（当時としては）非常に巧妙にできていて，（漢字の使えなかった）旧来のシステムとの整合性

[*9] RFC2046（http://www.asahi-net.or.jp/~bd9y-ktu/dtd_f/rfc_f/rfc2046j.html 参照）。

もよく，日本国内で急速に広まっていきました。しかし文字集合を無理矢理に詰め込んでいるため，拡張性に乏しいのが難点です。実際に補助漢字である JIS X 0212 はシフト JIS に取り込まれませんでした。なお新しくできた JIS X 0213 では，JIS X 0212 の反省を踏まえ，シフト JIS 用のエンコーディング（Shift_JISX0213）も提供しています。

シフト JIS が今後の国際化・多言語化に対応しきれないのは明らかであり，新しいコード体系へのシフトが求められています。

2.5 ISO/IEC 10646-1（Unicode）

1984 年，ISO は文字コードの国際化（I18N: InternationalizatioN）を図るため，全世界の主要な文字を含んだ単一の符号化文字集合 ISO/IEC 10646 の策定を開始しました。この作業が始まった同じ頃米国の有力なコンピュータ企業が集まって同じような符号化文字集合「Unicode[*10]」を開発しました。最終的には，両者の歩み寄りにより，「ISO/IEC 10646-1」に統合されました。日本では JIS X 0221 で定義されています。

ISO/IEC 10646-1 の符号化文字集合のことを特に「UCS」（Universal Multiple-Octet Coded character Set）と呼びます。またエンコーディングのことを「UTF」（UCS Transfer Format）と呼びます。

2.5.1 UCS

UCS には現在「UCS-2」と「UCS-4」があります。

UCS-4 は ISO/IEC 10646-1 が定義するフルサイズ（31 bit）のコード空間で群（group）・面（plane）・区（row）・点（cell）の 4 次元構成になっています。256 区 ×256 点で 1 つの面を構成し，更に 256 面で 1 つの群を構成しています。群は 128 あります。

00 群 00 面を特に「BMP」（Basic Multilingual Plane: 基本多言語面）と呼んでいます。現時点で定義されている文字集合は全て BMP に収められており，他の群・面には何も定義されていません。この BMP を 16 bit で符号化したものが UCS-2 です。（もともとの Unicode は UCS-2 のみ定義していました。従って Unicode は ISO/IEC 10646-1 のサブセットであるということもできます）

2.5.2 UTF

現在よく使われる UTF としては「UTF-16」，「UTF-8」，「UTF-7」があります。

UTF-16 は基本的には UCS-2 の符号をそのままコードとして使っています。ただし拡張用に，BMP に続く 16 面分を埋めこむことのできる「Surrogate Pair」という仕組みを持っています。

UTF-8 は UCS を ISO/IEC 2022 と整合性のある方法でエンコードしたものです。すなわち，UCS の IRV に相当する文字集合を GL に，残りの文字を 2〜6 byte の可変サイズの 96^n 文字集合として GR に配置します。従来のエンコーディングとの（特に GL において）衝突が少ないため，UTF-8 は広く受け入れられています。特にオープンソース・コミュニティでは事実上の国際標準コードとなりつつあります。

UTF-7 は UTF-8 を（電子メールのような）7 bit 伝送系でも通せるように（Base64 のような方法で）変換したものです。一般にはあまり使われませんが，8 bit データを扱えない特殊な条件下では使われることがあります。

[*10] http://www.unicode.org/ 参照。

3　文字コードにまつわる諸問題

日本では符号化文字集合は JIS で統一されているかのように見えますが，互換性などにおいて様々な問題を抱えています。これに加え，多様な文字エンコーディングが更に問題を複雑にしています。この章ではそういった問題点を具体的な事例を示しながら紹介していきます。

3.1　IRV と JIS ラテン文字

前述（2.1.1 章）したとおり，IRV と JIS ラテン文字の違いはわずか 2 字です。これは日本国内に限ってはあまり問題がないように思えますが，「国際化」を考える際には大きな問題になります。

まず「~」（TILDE）と「¯」（OVER LINE）ですが，JIS X 0201 の附属書 2 では（送受信者間で合意があれば）両者を区別しないとしています。Windows 等では日本語環境でも 7EH が「~」になっています。しかし一方で，ISO/IEC 10646-1 では両者は厳密に区別されています。

現実には「~」や「¯」を単独で意味ある文字として使うことはまずありません。「~」や「¯」は制御記号として使われることが多く，それ以外では合成文字の要素（「ã」の「~」など）として使われる程度です。しかも昨今では合成文字は（互換性の問題から）殆ど使われませんし，そういう意味では字形にこだわる必要はないのかもしれません。しかし，ISO/IEC 10646-1 で明確に区別されている以上どちらかを選ぶ必要があり，その「選択」を巡って混乱が起きることは必至です。

「\」（REVERSE SOLIDUS）と「¥」（YEN SIGN）では更に深刻です。例えば C 言語のプログラミングで日本円で金額を表示したい時，UCS を使ってコーディングするなら次のようになる筈です。

```
コーディング:  printf("Total:  ¥%d.\n", maney);
    出力例:  Total:  ¥646.
```

しかし，（UCS を使わない）これまでのコードはこのように区別されていません。JIS ラテン文字を実装する日本語圏のシステムなら次のようにコーディングされているでしょう。

```
コーディング:  printf("Total:  ¥¥%d.¥n", maney);
    出力例:  Total:  ¥646.
```

このコードを JIS ラテン文字を実装していない，例えば IRV のみのシステムに適用すると，出力結果は本来意図したものとは違うものになる筈です。

```
コーディング:  printf("Total:  \\%d.\n", maney);
    出力例:  Total:  \646.
```

国際化の流れの中では JIS ラテン文字は使われなくなる傾向にあります。（例えば ISO-2022-JP-3 では JIS ラテン文字は定義されていません）従って今後 JIS ラテン文字固有の文字（「¥」「¯」）を意味ある文字として使う際は，何らかの代替え手段（JIS 漢字集合を用いるか文字集合全体を UCS/UTF でエンコードする）を講じる必要があります。

3.2 「半角カナ」について

JIS X 0201 片仮名のことを俗に「半角カナ」といいます。初期のキャラクタベース端末で表示されるデザインからそう呼ばれているようです。日本のコンピュータ初期の時代では広く使われていましたが，これも最近の国際化の流れの中で使われなくなりつつあります。

まずインターネットの黎明期における混乱が発端といえるでしょう。当時インターネットの国内の前身である JUNET は UNIX 系プラットフォームをメインに使っていましたが，当時のシステムは「半角カナ」を実装するものが少なかったといわれています。（2.3.4 章参照）更に ISO に「半角カナ」の指示コードを申請する際に，当時は「H」で申請していたそうですが結局このコードは別のセットになり（表 7 参照），「半角カナ」は別のコード「I」に決まりました。しかし日本国内ではこのことによる混乱が収まらず，結局 ISO-2022-JP では「半角カナ」を使ってはいけないことに決まってしまいました。

JIS X 0208 附属書 1 ではシフト JIS の「半角カナ」を将来削除する予定であると謳っていて，JIS X 0213 でも「JIS X 0201 片仮名は原則として使わない」ことになっています。（ISO-2022-JP-3 では「半角カナ」は定義されていません）

JIS X 0201 のラテン文字および片仮名を総称して「半角文字」と呼ぶことがあります。これに対し JIS X 0208,0212,0213 の漢字集合を「全角文字」と呼びます。「全角文字」集合は「半角文字」の字形を含んでいて，これが特に国内において混乱の元になることがあります。更に ISO/IEC 10646-1 ではこの「通称」に配慮して「Halfwidth and Fullwidth Forms」という互換用の領域を作ってしまいました。これには「半角カナ」と「全角英数字」が収められていて，しかも JIS ラテン文字にない ISO/IEC 646 の各国語の文字の一部まで「全角英数字」として収容されています。

半角/全角の問題は，特に ISO/IEC 10646-1 も含めて考えると将来深刻な混乱を引き起こしかねません。メッセージシステムを設計・運用する際には相当な注意が必要です。なお「半角」と「全角」で字形が重複するものについては，JIS の各規格において以下の指針が与えられています。

- 「半角カナ」は使わず，「全角文字」で代替えする
- 「半角カナ」は漢字集合を含んだエンコーディングとしては将来廃止される。
- 「英数字」において IRV で定義される文字と同一のものについては「全角文字」を使用しない

3.3 旧 JIS 漢字と新 JIS 漢字

JIS X 0208 は 1983 年に大改定されました。1983 年以前の JIS 漢字集合（JIS C 6226-1978[*11]）を「旧 JIS」，1983 年以降の JIS 漢字集合（JIS X 0208,0212,0213）を「新 JIS」と呼びます。この大改定は主に字体や字形の変更や交換を行うものでしたが，コンピュータ界に大きな混乱をもたらしました。現在は殆どのプラットフォームで新 JIS への移行が完了していますが，これにより逆に旧 JIS で書かれたテキストが「文字化け」することになってしまいました。

最も顕著な例は ISO-2022-JP です。ISO-2022-JP では旧 JIS を許容していて旧 JIS と新 JIS の混在するテキストを書くことも可能です。しかし旧 JIS のセットを持っていない処理系では，旧 JIS 部分も無理矢理新 JIS で表示せざるをえません。また ISO-2022-JP 以外のエンコーディングでは旧 JIS と新 JIS を区別できないた

[*11] JIS X 0208 はそれまで JIS C 6226 と呼ばれていました。

め，ISO-2022-JP から他のエンコーディングに変換する際に旧 JIS の情報が失われてしまいます。

このような混乱を回避するため，特に電子メールにおいては，以下のようなローカルルールがあります。

- JIS 漢字集合を使う際は新 JIS を指示（1BH 24H 42H）する
- 1 byte 文字を使う際は IRV を指示（1BH 28H 42H）する
- JIS ラテン文字および旧 JIS は使用しない

これは RFC1468 には記載されていませんが，JIS 漢字を含むメールをやり取りを行う際の暗黙のルールになっています。

3.4 ベンダ固有文字

最初の JIS X 0208（JIS C 6226-1978）以前はメーカごとに文字コードを策定していました。JIS 漢字制定後もそれまでの文字コードとの互換性を確保するため，JIS 漢字の不足分を独自に追加しています。これが「ベンダ固有文字」です。「機種依存文字」とか「システム外字」とか呼ばれることもあります。ベンダ固有文字としては以下の 2 つが有名です。

IBM 拡張文字 ：
: IBM が自社のメインフレームの文字集合から JIS C 6226-1978 でカバーされない文字を選んだもの。

NEC 拡張文字 ：
: NEC が IBM 拡張文字に加えてさらに記号類や半角文字を追加したもの。俗に言う「98 文字」。

これらの文字は JIS C 6226-1978 の符号化文字集合の空き領域（1 面の 9〜15 区・85〜94 区および 2 面[*12]）に配置されていて，JIS C 6226-1978 で収容されなかった記号や漢字の異体字などが収められています。

ベンダ固有文字の定義はベンダやプラットフォームごとに違っていて，殆ど互換性がありません。JIS 規格でも情報交換用としてはベンダ固有文字などの JIS 未定義文字を使うことを禁じています。しかし，ユーザの立場では使っている文字がベンダ固有文字かどうか（特に異体字は）見た目でわかりにくく，しばしば混乱の元になっています。

JIS X 0213 ではこれらのベンダ固有文字を含めた形で符号化文字集合が定義されています。異体字についても包摂[*13] の例外としてかなりの数が新たに追加されているようです。今後 JIS X 0213 がどの程度普及するか分かりませんが（現在 UNIX 系プラットフォームを中心に整備されつつあります），JIS X 0213 と情報交換用のエンコーディングである ISO-2022-JP-3 が定着すれば，ベンダ固有文字に対する考え方が変わるかもしれません。

一方 ISO/IEC 10646-1 ではベンダ固有文字および JIS X 0213 漢字集合の殆どが既に収容されています[*14]。したがって，電子メールなどでもエンコードを UTF-8/UTF-7 とすれば，ベンダ固有文字と言われていた文字も使うことができます。

[*12] シフト JIS では 2 byte 文字の第 1 byte が 85H〜88H・EBH〜FFH の範囲にある領域にあたります。

[*13] JIS 漢字には「包摂規準」というのがあり，意味が同じで似た字形の漢字を「同じ字体」とみなすことになっています。異体字の殆どはあるひとつの字体に包摂されるため，JIS 漢字として収容されません。

[*14] JIS X 0213 漢字集合のうち ISO/IEC 10646-1 に収容されていない文字は 360 字程あるそうです。

4　おわりに

本テキストでは「文字コード」の問題について一通り紹介しました。実際には「文字」や「文字コード」の問題はこれだけではなく，漢字の包摂と異体字の問題や Unicode における「CJK 統合」問題など色々ありますが，「情報交換」において問題になりそうなところを中心に挙げていきました。今回は問題を提示するだけでしたが，電子メールなど異なるプラットフォーム間でのメッセージシステムを設計・運用する際の参考になると思います。

なお，文字コードについての文献を末尾に挙げていますが，インターネット上でもいくつか情報があります。主なものを以下に挙げます。是非参考にしてください。

- 「文字コードの世界」（http://euc.jp/i18n/charcode.ja.html）
- 「従来の文字コードと Unicode の対応に関する諸問題」（http://euc.jp/i18n/ucsnote.ja.html）
- 「JIS と ISO-2022」（http://www.d2.dion.ne.jp/~imady/kcode/kcode_jis.html）
- 「Mew ニュースレター: ASCII」（http://www.mew.org/Newsletters/6.html）
- 「JIS X 0213 の代表的な符号化方式」
  （http://www.asahi-net.or.jp/~wq6k-yn/code/enc-x0213.html）

参考文献

[1] 日本規格協会（編）. JIS ハンドブック 64 情報技術 I（用語/符号/データコード）. 日本規格協会, 2001.
[2] 清水哲郎. 図解でわかる 文字コードのすべて. 日本実業出版社, 2001.
[3] 安岡孝一, 安岡素子. 文字コードの世界. 東京電機大学出版局, 1999.