原稿

１０章 情報の表現 ビットバイトコード情報量（１５）

1. 情報とデータ（1）

●情報とは，次に何がくるかわからないときに，次に何かがくることにより得られる確かさ．

情報の単位は，二者択一の事象の 1 つが起れば，1 ビットの情報四者択一の事象の 1 つが起れば，2 ビットの情報生起確立が等しい Q 個の事象の 1 つが起こったことにより得られる情報は $log_2 Q$ ビット

事象 E の生起確立が $P(E)$ である場合，E が起こったことにより得られる情報は $-log_2 P(E)$ ビット

●データとは，情報を計算機が処理可能な形にしたもの．記号による表現．実世界の対象物をわかりやすく，少ない量で表現できること．現実の計算機の上で，効率良く処理できること．

「情報の構造の明確化」＝「記号の性質および記号間の関係の明確化」に置き換えることができる．

●情報の最小単位：ビット

情報の最小単位はビットと呼ばれる．1 ビットは，0 または 1 のいずれかを記憶することのできる箱であり，晴れか雨，表か裏，○か×といった 2 つの状態のいずれであるかを表示することができる．1 ビットを 2 個用意すると，00，01，10，11 のいずれかを記憶することができ，晴れ/雨/曇り/雪といった 4 通りの状態に対応することができる．さらに 1 ビットを 8 個用意したものは 1 バイトと呼ばれ，２５６通りの情報を表示することができる．

・1ビット　□ 0または1の2通り
・2ビット　□□ 00 01 10 11の4通り
・1バイト(8ビット)　□□□□□□□□ 2の8乗=256通り

これに対しワードという単位は，コンピュータの機種によって異なる．1 ワードは，16 ビットコンピュータでは 16 ビットとなり，32 ビットコンピュータでは 32 ビットとなる．

・1ワード(16ビット)　2の16乗=6万5536通り
・1ワード(32ビット)　2の32乗=42億9496万7296通り
・1ワード(64ビット)　2の64乗=1844京6744兆…通り

2. 基本的なデータ型（３）

2.1 整　　数

ビット数が同じであっても，型によって表現できる数の範囲が異なる．例えば８ビットの場合，符号付き整数，符号無し整数，浮動小数点数は，それぞれ以下のように異なる．ただし，８ビット浮動小数点数は現実には存在しない．３２ビット以上の浮動小数点数について規定している IEEE754 の考え方を仮に８ビットに対して適用した場合の表現である．

2.1.1 符号無し整数 … ８ビット長の場合

```
00000000　　…　　0
00000001　　…　　1
00000010　　…　　2
00000011　　…　　3
11111111　　… ２５５
```

演算

```
00000010（　　2）＋ 00000011（3）＝ 00000101（　　5）
```

オーバフロー/アンダフロー

```
11111111（２５５）＋ 00000011（3）＝ 00000010（　　2）
00000010（　　2）－ 00000011（3）＝ 11111111（２５５）
```

2.1.2 符号付き整数 … ８ビット長の場合

```
10000000　　… －１２８
11111111　　… －　　1
00000000　　…　　0
00000001　　… ＋　　1
01111111　　… ＋１２７
```

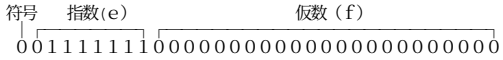
演算

```
11111111（　　－1）＋ 00000011（3）＝ 00000010（　　2）
```

オーバフロー/アンダフロー

```
01111111（　１２７）＋ 00000011（3）＝ 10000010（－１２６）
10000010（－１２６）－ 00000011（3）＝ 01111111（　１２７）
```

2.2 浮動小数点数



2.2.1 単精度浮動小数点数 … ３２ビット長

単精度浮動小数点数は float 型に対応している．IEEE754 形式の場合，符号 1 ビット，指数部（ｅ）8 ビット，仮数部（ｆ）23 ビットの合計 32 ビットから構成されている．符号が０の場合正数，１の場合負数を表す．

符号付きゼロ $e = 0, f = 00\cdots00$ の場合，0.0 を表す．符号と組み合わせて +0

と -0 の 2 通りがある．

不正規化数 $e = 0, f \neq 00\cdots00$ の場合は不正規化数と呼び，$2^{-126} * 0.f$ を表す．$e = 0, f = 00\cdots01$ の場合，$1.40129846 * 10^{-45}$（0 に最も近い値），$e = 0, f = 11\cdots11$ の場合，$1.17549421 * 10^{-38}$（最大の不正規化数）．

正規化数 $0 < e < 255$ の場合は正規化数と呼び，$2^{e-127} * 1.f$ を表す．$e = 1, f = 00\cdots00$ の場合，$1.17549435 * 10^{-38}$（最小の正規化数），$e = 127, f = 00\cdots00$ の場合，1.00000000．

無限大 $e =$ 最大 $(255), f = 00\cdots00$ の場合は無限大を表す．符号と組み合わせて $+\infty$ と $-\infty$ の 2 通りがある．

非数値 $e =$ 最大 $(255), f \neq 00\cdots00$ の場合は非数値を表す．符号は意味を持たない．例えば 0 ÷ 0 の結果が相当する．

例えば，円周率（3.14159265358979323846…）は，単精度浮動小数点数を用いた場合，以下の★に示すように 3.141592 までは正しく表現することができる．

```
符号 指数部(e)        仮数部(f)
 0/1 00000000 00000000000000000000000  ⇒ +/-0
 0/1 01111100 00000000000000000000000  ⇒ +/-0.125
 0/1 01111101 00000000000000000000000  ⇒ +/-0.25
 0/1 01111110 00000000000000000000000  ⇒ +/-0.5
 0/1 01111111 00000000000000000000000  ⇒ +/-1
 0/1 10000000 00000000000000000000000  ⇒ +/-2
 0/1 10000000 10000000000000000000000  ⇒ +/-3
 0/1 10000000 10010010000111111011010  ⇒ +/-3.14159250
 0/1 10000000 10010010000111111011011  ⇒ +/-3.14159274 ★円周率に近い値
 0/1 10000000 10010010000111111011100  ⇒ +/-3.14159297
 0/1 10000000 11111111111111111111111  ⇒ +/-3.99999976
 0/1 10000001 00000000000000000000000  ⇒ +/-4
 0/1 10000010 00000000000000000000000  ⇒ +/-8
 0/1 11111111 00000000000000000000000  ⇒ +/-∞
 0/1 11111111 xxxxxxxxxxxxxxxxxxxxxxx  ⇒ 非数値（xxxxは0000以外）
```

図 1　単精度浮動小数点数による表現

以上のように，浮動小数点数と言えども，任意の数値を表現することはできず，離散的な数値しか表現できないことがわかる．

2.2.2　倍精度浮動小数点数

同様に，倍精度浮動小数点数は double 型に対応している．IEEE754 形式の場合，符号 1 ビット，指数部（e）11 ビット，仮数部（f）52 ビットの合計 64 ビットから構成されている．符号が 0 の場合正数，1 の場合負数を表す．

符号付きゼロ $e = 0, f = 00\cdots00$ の場合，0.0 を表す．符号と組み合わせて $+0$ と -0 の 2 通りがある．

不正規化数 $e = 0, f \neq 00\cdots00$ の場合は不正規化数と呼び，$2^{-1022} * 0.f$ を表す．$e = 0, f = 00\cdots01$ の場合，$4.9406564584124654 * 10^{-324}$ （0 に最も近い値）．$e = 0, f = 11\cdots11$ の場合，$2.2250738585072008 * 10^{-308}$（最大の不正規化数）．

正規化数 $0 < e < 2047$ の場合は正規化数と呼び，$2^{e-1023} * 1.f$ を表す．$e = 1, f = 00\cdots00$ の場合，$2.2250738585072013 * 10^{-308}$（最小の正規化数）．$e = 1023, f = 00\cdots00$ の場合，1.0000000000000000．

無限大 $e =$ 最大 (2047), $f = 00\cdots00$ の場合は無限大を表す．符号と組み合わせて $+\infty$ と $-\infty$ の 2 通りがある．

非数値 $e =$ 最大 (2047), $f \neq 00\cdots00$ の場合は非数値を表す．符号は意味を持たない．例えば 0 ÷ 0 の結果が相当する．

例えば，円周率（3.14159265358979323846…）は，倍精度浮動小数点数を用いた場合，以下の★に示すように 3.141592653589793 までは正しく表現することができる．

```
符号  指数部(e)              仮数部(f)
 0/1 00000000000 0000000000000000000000000000…0000 ⇒ +/-0
 0/1 01111111100 0000000000000000000000000000…0000 ⇒ +/-0.125
 0/1 01111111101 0000000000000000000000000000…0000 ⇒ +/-0.25
 0/1 01111111110 0000000000000000000000000000…0000 ⇒ +/-0.5
 0/1 01111111111 0000000000000000000000000000…0000 ⇒ +/-1
 0/1 10000000000 0000000000000000000000000000…0000 ⇒ +/-2
 0/1 10000000000 1000000000000000000000000000…0000 ⇒ +/-3
 0/1 10000000000 1001001000011111101101010…0111 ⇒ +/-3.14159265358979267
 0/1 10000000000 1001001000011111101101010…1000 ⇒ +/-3.14159265358979311 ★円周率に近い値
 0/1 10000000000 1001001000011111101101010…1001 ⇒ +/-3.14159265358979356
 0/1 10000000000 1111111111111111111111111111…1111 ⇒ +/-3.99999999999999955
 0/1 10000000001 0000000000000000000000000000…0000 ⇒ +/-4
 0/1 10000000010 0000000000000000000000000000…0000 ⇒ +/-8
 0/1 11111111111 0000000000000000000000000000…0000 ⇒ +/-∞
 0/1 11111111111 XXXXXXXXXXXXXXXXXXXXXXXXXXXX…XXXX ⇒ 非数値
```

図 2　倍精度浮動小数点数による表現

このように，コンピュータの内部形式では，任意の浮動小数点数を正確に表現することはできない．浮動小数点数を有限桁に丸めたために発生する誤差を「丸め誤差」と呼ぶ．

○拡張倍精度浮動小数点数もある（符号 1 指数 15 仮数 64 または 符号 1 指数 15 仮数 112）

実数/浮動小数点数（VAX 形式）

実数/浮動小数点数（IBM 形式）

2.3 論 理 値

1ビット長の場合

```
0          … 偽（FALSE）
1          … 真（TRUE）
```

8ビット長の場合

```
00000000   … 偽（FALSE）
その他     … 真（TRUE）
```

各ビットを独立論理値とした場合のビット毎演算

```
否定：                NOT  00001111 ＝ 11110000
論理和：    00111100  OR   00001111 ＝ 00111111
論理積：    00111100  AND  00001111 ＝ 00001100
排他論理和：00111100  XOR  00001111 ＝ 00110011
```

実際には，ワード単位で扱うほうが，１ビットだけを扱うよりも手間が少ない.

2.4 文　　字

null 文字 00000000 … ” １バイト文字 (半角文字) 00110010 … ’2’ ２バイト文字 (全角文字) 10100011,10110010 … ’２’ 文字の演算（比較）が可能 00110010（’2’）＜ 00110011（’3’）10100100,10100010（’あ’）＜ 10100100,11110011（’ん’）

2.5 文 字 列

文字の並び＋終端 (null) 文字 00110010,00110011,00000000 … ”23”

空の文字列 00000000 … ””

文字列の演算（比較）が可能”あい”＜”あん”

3. エンディアン問題（1）

・Big/little endian のところは，おなじ 32 ビットデータでも，よりワードアクセスらしい，整数値で説明する方が適当ではないでしょうか？

データの桁位置の高低下位ビット … 桁位置が低いビット最下位ビット＝ Least Significant Bit

上位ビット … 桁位置が高いビット最上位ビット＝ Most Significant Bit

主記憶アドレスの高低低アドレス … 数値として小（0x00010000）高アドレス … 数値として大（0xffff000）

3.1 ビッグエンディアン

… IBM 汎用機，SPARC など

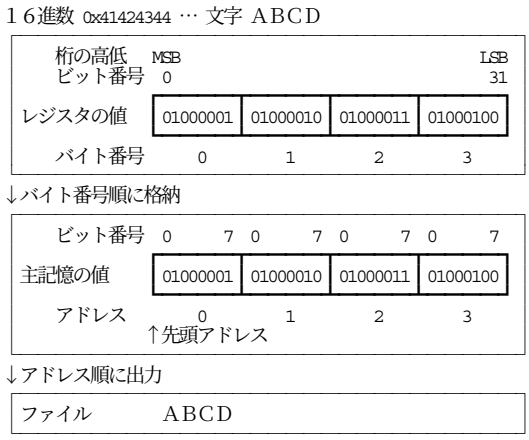


図 3 ビッグエンディアン

3.2 リトルエンディアン

… Pentium など

エンディアンの異なる計算機間では，主記憶データに互換性がない．C 言語の read/write などにより，主記憶データを直接ファイルに格納したもの．構造体をファイルに格納する場合は，個々の要素のエンディアンを考慮しなければならな

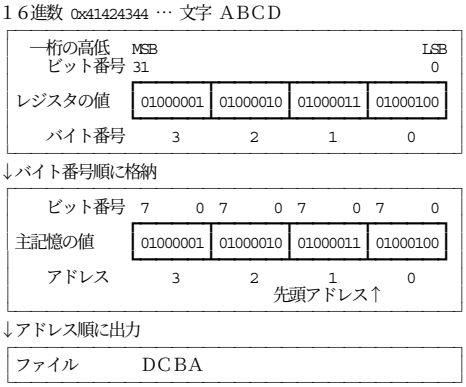


図 4 リトルエンディアン

い．ネットワーク間でデータを共有する場合にはエンディアンに関する取り決めが必要．移植性の高いソフトウェアを開発する際も同様．

4. 文字コード（4）

4.1 EBCDIC

… IBM 汎用機

上位4ビット＼下位4ビット	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	PF	HT	LC	DEL	GE	RLF	SMM	VT	FF	CR	SO	SI
1	DLE	DC1	DC3	TM	RES	NL	BS	IL	CAN	EM	CC	CU1	IFS	IGS	IRS	IUS
2	DS	SOS	FS		BYP	LF	ETB	ESC			SM	CU2		ENQ	ACK	BEL
3			SYN		PN	RS	UC	EOT				CU3	DC4	NAK		SUB
4	SP										¢	.	<	(	+	|
5	&										!	$	*	)	;	
6	−	／										,	%	_	>	?
7											:	#	@	’	=	”
8		a	b	c	d	e	f	g	h	i						
9		j	k	l	m	n	o	p	q	r						
A			s	t	u	v	w	x	y	z						
B																
C		A	B	C	D	E	F	G	H	I						
D		J	K	L	M	N	O	P	Q	R						
E			S	T	U	V	W	X	Y	Z						
F	0	1	2	3	4	5	6	7	8	9	LVM					EO

図 5　EBCDIC

4.2 EBCDIK

… 片仮名に対応

4.3 ASCII

4.4 JIS X0201 ローマ字

4.5 制御コード

4.6 JIS X0201 片仮名

いわゆる半角片仮名，使用は避けるべき

・文字コード表の中の『カタカナ』は，見た目には「全角文字」になっていま

（下位4ビット）

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	PF	HT	LC	DEL	GE	RLF	SMM	VT	FF	CR	SO	SI
1	DLE	DC1	DC3	TM	RES	NL	BS	IL	CAN	EM	CC	CU1	IFS	IGS	IRS	IUS
2	DS	SOS	FS		BYP	LF	ETB	ESC			SM	CU2		ENQ	ACK	BEL
3			SYN		PN	RS	UC	EOT				CU3	DC4	NAK		SUB
4	SP	。	「	」	、	・	ヲ	ァ	ィ	ゥ	[	.	<	(	+	|
5	&	ェ	ォ	ャ	ュ	ョ	ッ		ー		]	¥	*	)	;	
6	−	／									^	,	%	_	>	?
7											:	#	@	’	=	”
8		ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ		サ	シ	ス	セ
9	ソ	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ			ハ	ヒ	フ
A			ヘ	ホ	マ	ミ	ム	メ	モ	ヤ	ユ		ヨ	ラ	リ	ル
B											レ	ロ	ワ	ン	゛	゜
C	{	A	B	C	D	E	F	G	H	I						
D	}	J	K	L	M	N	O	P	Q	R						
E	$		S	T	U	V	W	X	Y	Z						
F	0	1	2	3	4	5	6	7	8	9	LVM					EO

図 6　EBCDIK

（下位4ビット）

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	NL	VT	NP	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	”	#	$	%	&	’	(	)	*	+	,	−	.	／
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	＼	]	^	_
6	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	～	DEL

図 7　ASCII

すよね。もし，このような表を載せるなら，印刷時に「半角文字」に見えるように注意する必要があるかもしれません。（図 10.14 あたりは，個々のカ　タ

（下位4ビット）

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	NL	VT	NP	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	”	#	$	%	&	’	(	)	*	+	,	−	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	¥	]	^	_
6	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	¯	DEL

図 8　JIS X0201 ローマ字

（下位4ビット）

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	NL	VT	NP	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP															
7																DEL

● NUL：NULl
　SOH：Start Of Heading
　STX：Start of TeXt
　ETX：End of TeXt
● EOT：End Of Transmission
　ENQ：ENQuiry
　ACK：ACKnowledge
● BEL：BELl
● BS　：BackSpace
● HT　：Horizontal Tab
● NL　：New Line
　VT　：Vertical Tab
● NP　：New Page
● CR　：Carriage Return
　SO　：Shift Out
　SI　：Shift In
　DLE：Data Link Escape
　DCn：Device Control n
　NAK：Negative AcKnowledge
　SYN：SYNchronous idle
　ETB：End of Transmission Block
　CAN：CANcel
　EM　：End of Medium
　SUB：SUBstitute
● ESC：ESCape
　FS　：File Separator
　GS　：Group Separator
　RS　：Record Separator
　US　：Unit Separator
● SP　：SPace
● DEL：DELete

図 9　制御コード

カナを表示しないで，横抜きで「半角カナ」としてしまった方がいいかも....）

4.7　JIS C6226-1978（旧ＪＩＳ）

…日本規格協会「第 1 水準」2965 字,「第 2 水準」3384 字,「非漢字」453 字

4.8　JIS X0208-1983（新ＪＩＳ）

…日本規格協会「第 1 水準」2965 字,「第 2 水準」3388 字,「非漢字」524 字

（下位4ビット）

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		。	「	」	、	・	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ
B	ー	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ
C	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ
D	ミ	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ン	゛	゜

図 10　JIS X0201 片仮名

図 11　JIS C6226-1978（旧ＪＩＳ）

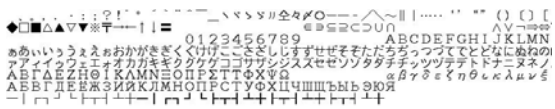


図 12　JIS X0208-1983（新ＪＩＳ）

4.9　JIS X0212-1990（補助漢字）

「漢字」5801 字,「非漢字」266 字

4.10　ISO 10646, JIS X0221-1995（ＵＣＳ）

…日本規格協会

UCS-4 ： 1 文字を 4 バイトで表現上位 2 バイトが 0 の範囲を BMP（Basic Multilingual Plane）と呼ぶ.

UCS-2：BMP の上位 2 バイトを省いた 2 バイトコードこれがいわゆる Unicode

図 13　JIS X0212-1990（補助漢字）

5. 日本語コード（4）

5.1 シフトＪＩＳコード

… Microsoft，アスキー特別な遷移コードがない代わりに字数が少ない

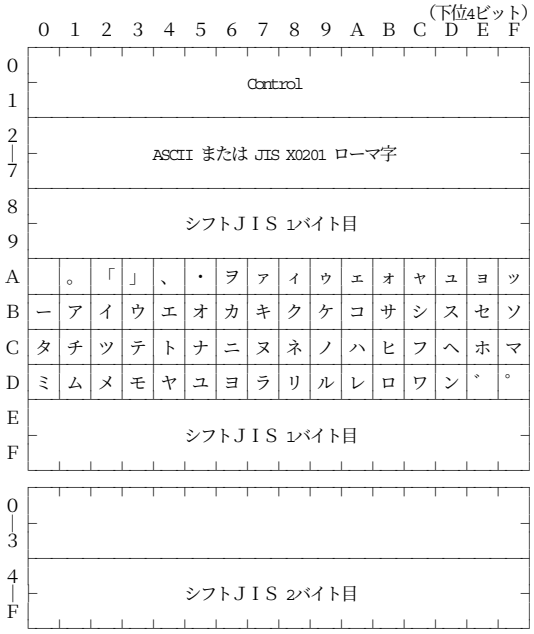


図 14　シフトＪＩＳコード

5.2 ISO-2022-JP

（いわゆるＪＩＳコード）7 ビットの文字列で漢字を表現できる

2 バイト文字への遷移コード JIS C6226-1978（旧 JIS）へは，1B2440 ESC@$JISX$0208−1983（新 JIS）へは，1B2442ESCB

1 バイト文字への遷移コード

JIS X0201-1976 ローマ字へは，1B284A ESC(J ASCII へは，1B2842 ESC(B

ISO-2022-JP には JIS X0201 片仮名が無い．

JIS7 方式 … 20〜5F に割り当てる．1B2849 ESC(I を片仮名開始とする方法

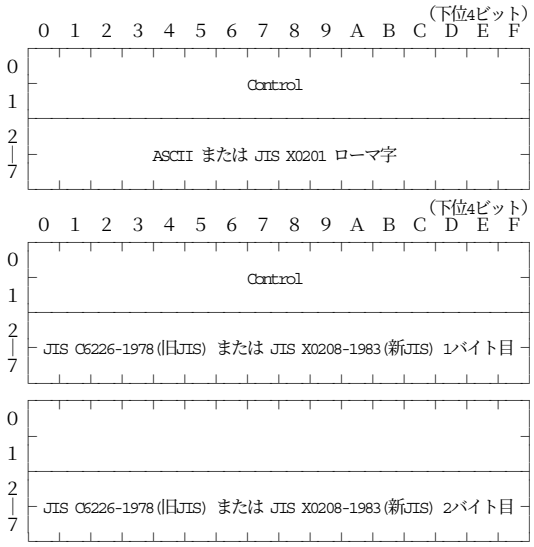


図 15　ISO-2022-JP（いわゆるＪＩＳコード）

0E/0F（ShiftOut/In）を片仮名開始/終了とする方法

JIS8 方式 … A1〜DF に割り当てる.

5.3　NEC 漢字コード

独自の遷移コード 1B4B/1B48（KanjiIn/Out）

5.4　EUC（Extended Unix Code）

重複する場合，シフト JIS と区別できない

5.5　UTF-8

UCS-2 に基づく 1〜3 バイトの文字コード

0000〜007F の場合下 1 バイトを使用

0080〜07FF の場合（00000xxxxxyyyyyy）110xxxxx を 1 バイト目，10yyyyyy を 2 バイト目とする

0800〜FFFF の場合（xxxxyyyyyyzzzzzz）1110xxxx を 1 バイト目，10yyyyyy を 2 バイト目，10zzzzzz を 3 バイト目とする

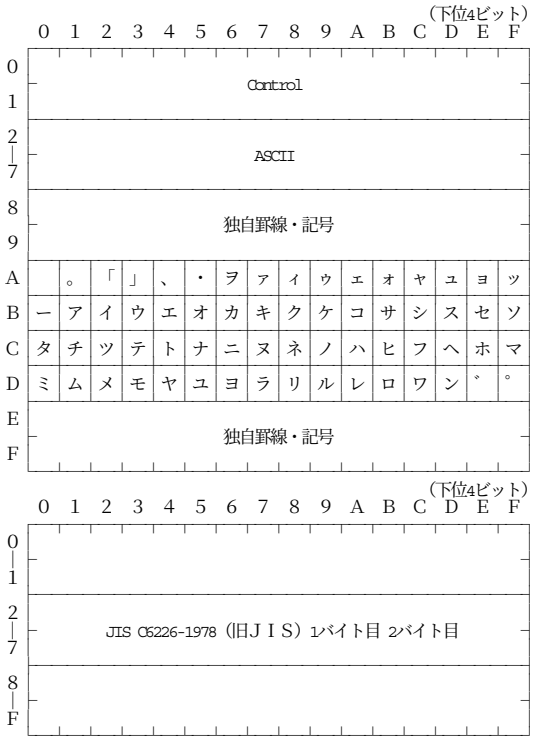


図 16　NEC 漢字コード

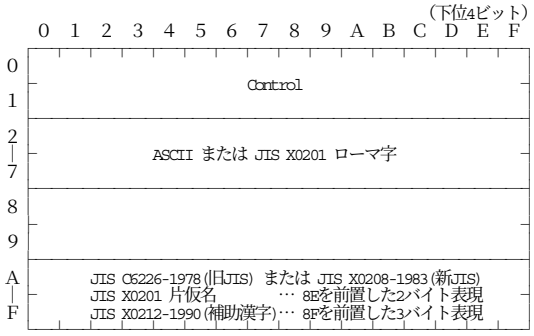


図 17　EUC（Extended Unix Code）