地球流体電脳ライブラリ

MISC1

(その他の基本処理下位パッケージ)

地球流体電脳倶楽部

1995 年 6 月 9 日 (DCL-5.0)

目 次

1 概要 **1**
1.1 はじめに . . . 1
1.2 整数の内部表現 . . . 3
1.3 実数の内部表現 . . . 5
1.4 文字の内部表現 . . . 8
1.5 ファイルの構造 . . . 13
1.5.1 メインフレーム系 . . . 13
1.5.2 UNIX 系 . . . 14

2 BITLIB：ビットパターンの処理 *☆ **15**
2.1 概要 . . . 15
2.2 サブルーチンのリスト . . . 15
2.3 関数のリスト . . . 15
2.4 サブルーチンの説明 . . . 15
2.4.1 BITPIC . . . 15
2.4.2 BITPCI . . . 16
2.4.3 GBYTE/GBYTES . . . 16
2.4.4 SBYTE/SBYTES . . . 17
2.5 関数の説明 . . . 17
2.5.1 ISHIFT . . . 17
2.5.2 IAND/IOR . . . 18

3 CHGLIB：大文字・小文字の変換 *☆ **19**
3.1 概要 . . . 19
3.2 サブルーチンのリスト . . . 19
3.3 サブルーチンの説明 . . . 19
3.3.1 CUPPER/CLOWER . . . 19

4 CHKLIB：文字種の判別 * **20**
4.1 概要 . . . 20
4.2 関数のリスト . . . 20
4.3 関数の説明 . . . 21
4.3.1 LCHRB/LCHRC/LCHRS/LCHRL/LCHRD/LCHRA/LCHRF . . . 21
4.3.2 LCHR . . . 21

5 CHNLIB：文字列の置換 * **23**
5.1 概要 . . . 23
5.2 サブルーチンのリスト . . . 23
5.3 サブルーチンの説明 . . . 23
5.3.1 CHNGC . . . 23
5.3.2 CHNGI/CHNGR . . . 24

6 FMTLIB：数値の文字列化 * **25**
6.1 概要 . . . 25
6.2 サブルーチンのリスト . . . 25
6.3 サブルーチンの説明 . . . 25
6.3.1 CHVAL . . . 25

7 DATELIB：日付の取り扱い *☆ **27**
7.1 概要 . . . 27
7.2 サブルーチンのリスト . . . 27
7.3 関数のリスト . . . 28
7.4 サブルーチンの説明 . . . 28
7.4.1 DATE12/DATE13/DATE21/DATE23/DATE31/DATE32 . . . 28
7.4.2 DATEF1/DATEF2/DATEF3 . . . 29
7.4.3 DATEG1/DATEG2/DATEG3 . . . 29
7.4.4 DATEQ1/DATEQ2/DATEQ3 . . . 30
7.4.5 DATEC1/DATEC2/DATEC3 . . . 30
7.5 関数の説明 . . . 31
7.5.1 NDATE1/NDATE2/NDATE3 . . . 31
7.5.2 IWEEK1/IWEEK2/IWEEK3 . . . 32
7.5.3 NDMON/NDYEAR . . . 32
7.5.4 CMON . . . 33
7.5.5 CWEEK . . . 33

8 TIMELIB：時刻の取り扱い *☆ **35**
8.1 概要 . . . 35
8.2 サブルーチンのリスト . . . 35
8.3 サブルーチンの説明 . . . 35
8.3.1 TIME12/TIME13/TIME21/TIME23/TIME31/TIME32 . . . 35
8.3.2 TIMEQ1/TIMEQ2/TIMEQ3 . . . 36
8.3.3 TIMEC1/TIMEC2/TIMEC3 . . . 37

9 MISCLIB：雑多な関数 ･サブルーチン *☆ **38**
9.1 概要 . . . 38
9.2 サブルーチンのリスト . . . 38
9.3 関数のリスト . . . 38
9.4 サブルーチンの説明 . . . 38
9.4.1 DCLVNM . . . 38
9.4.2 CDBLK . . . 39

9.5 関数の説明 . . . 39
9.5.1 CSN . . . 39

10 CLCKLIB：CPU 時間の取り扱い ☆ 40
10.1 概要 . . . 40
10.2 サブルーチンのリスト . . . 40
10.3 サブルーチンの説明 . . . 40
10.3.1 CLCKST . . . 40
10.3.2 CLCKGT . . . 40
10.3.3 CLCKDT . . . 41

11 FIOLIB：ファイルの入出力 ☆ 42
11.1 概要 . . . 42
11.2 サブルーチンのリスト . . . 43
11.3 サブルーチンの説明 . . . 43
11.3.1 FCOPEN . . . 43
11.3.2 FCCLOS . . . 44
11.3.3 FCSLFC . . . 44
11.3.4 FCLEOL . . . 44
11.3.5 FCNREC . . . 45
11.3.6 FCGETR . . . 45
11.3.7 FCPUTR . . . 46
11.3.8 FCGETS . . . 46
11.3.9 FCPUTS . . . 46
11.3.10 FCRWND . . . 47

12 RANDLIB：疑似乱数 ☆ 48
12.1 概要 . . . 48
12.2 関数のリスト . . . 48
12.3 関数の説明 . . . 48
12.3.1 RNGU0/RNGU1/RNGU2/RNGU3 . . . 48

13 HEXLIB：16 進定数の処理 50
13.1 概要 . . . 50
13.2 サブルーチンのリスト . . . 50
13.3 サブルーチンの説明 . . . 50
13.3.1 HEXDIC . . . 50
13.3.2 HEXDCI . . . 51

14 REALLIB：実数の変換 52
14.1 概要 . . . 52
14.2 関数のリスト . . . 52
14.3 関数の説明 . . . 52
14.3.1 R4IBM . . . 52
14.3.2 R4IEEE . . . 53

謝辞 54

第 1 章

概要

1.1 はじめに

MISC1 (その他の基本処理下位パッケージ) は, ビットパターン ・文字の処理や, ファイルの入出力などシステムに依存するライブラリ, および日付 ・時間など数学的取り扱いとはやや異なるライブラリなどを含んでいる. 具体的には, 以下のようなライブラリからなる.

• **BITLIB:**	ビットパターンの処理.	*	☆	C
• **CHGLIB:**	大文字 ・小文字の変換.	*	☆	C
• **CHKLIB:**	文字種の判別.	*		
• **CHNLIB:**	文字列の置換.	*		
• **FMTLIB:**	数値の文字列化.	*		
• **DATELIB:**	日付の取り扱い.	*	☆	D(C)
• **TIMELIB:**	時刻の取り扱い.	*	☆	D(C)
• **MISCLIB:**	雑多な関数 ・サブルーチン.	*	☆	FC
• **CLCKLIB:**	CPU 時間の取り扱い.		☆	C
• **FIOLIB:**	ファイルの入出力.		☆	
• **RANDLIB:**	疑似乱数.		☆	FC
• **HEXLIB:**	16 進定数の処理.			
• **REALLIB:**	実数の変換.			

ここで *, ☆, F, C, D の各印は, 次のような意味を持つ.

***:** misc1 以外のライブラリの実行に必要なパッケージ.

☆: 機種依存性のあるサブルーチンを含む.

D : 機種依存性のあるサブルーチンをダミールーチンとしても, 致命的な支障はないもの.

F : 機種依存性のあるサブルーチンに対して, FORTRAN の プロトタイプ (機種依存性がないと思われるコード) が用意されているもの.

C : 機種依存性のあるサブルーチンに対して, C の 関数が用意されているもの.

詳しくは各節を参照されたい.

なお, 以下の節では, MISC1 パッケージが扱う内容を, 文字および数の計算機に内での表現, そして, 文字 ・数の集まりであるファイルの構造に照らして概説しておくことにする.

1.2 整数の内部表現

BITLIB はビット操作をするためのライブラリである. ビット操作をする変数には通常整数が使われる. その際に, 整数がどのようなビット列として表現されているかということが問題となる. また, ビット列を扱う場合には, 10 進の整数値として扱うより, 16 進数として扱った方が便利な場合もある. そのためのライブラリが HEXLIB である.

符号つきの整数を計算機の内部で表現する方法としては 「絶対値表示」と 「補数表示」がある. 「絶対値表示」とは整数の絶対値を 2 進数で表現して, その先頭ビットに符号ビット (正ならば 0, 負ならば 1) をつけたものである. この方法は素朴でわかりやすいが, 0 という整数の表現方法は 0 と -0 の 2 種類存在することになる.

「補数表示」は以下に示すように, もう少し込み入った定義になっているが, ほとんどの計算機では, この補数表示が採用されている.

- 補数表示

  32 ビットの符号つき整数の補数表示では, 0 及び正の整数は 「絶対値表示」同様に, 31 ビットの 2 進数の先頭に符号ビット (0) をつけたもので表される.

  これに対して, 負の整数は, その整数の絶対値の 「補数」に符号ビット (1) をつけたもので表される. 一般に複数の 「補数」の定義があるが, ここで言う補数とは 「2 進数における 2 の補数」(真補数) のことである. この場合, ある数 X の補数とは, 「X に加えると 2^n になるような数」で, 「各ビットの 0 と 1 を入れ換え, 最下位桁に 1 を足した数」といっても同じである.

  要するに, 符号つき整数を符号ビットまで含めて単純に 32 ビットの (正の)2 進数と見なした時, 負の整数 N は $2^{31}+N$ として表される (下表). この方法では, -2^{31} から $2^{31}-1$ までの数を表現できる.

内部表現	数値
11111111 11111111 11111111 11111111	-1
11111111 11111111 11111111 11111110	-2
.....	.
.....	.
10000000 00000000 00000000 00000000	-2^{31}
01111111 11111111 11111111 11111111	$+2^{31}-1$
.....	.
.....	.
00000000 00000000 00000000 00000001	1
00000000 00000000 00000000 00000000	0

  この表現を用いると, 加減算において符号部分も含めて正の数と統一的に扱うことができるようになり, 演算回路の構成が簡単になるため, 多くの計算機でこの表現法が用いられている.

このように補数表示では, 内部表現のビット列と整数としての値が 1 対 1 に対応しているので, データを数字ではなくビット列として処理したい時には, 整数型の変数が使われる.

1.3 実数の内部表現

整数の内部表現に関しては, ほとんどの機種で同一の表現となっているのに対して, 実数型に関してはいくつかの規格が存在する. このように実数表現形式の異なる計算機同志でデータを交換するためのユーティリティーが REALLIB である.

一般に, 実数型の変数は浮動小数点の形で表現される. すなわち, β進法を使った場合, 実数は

$$\pm\,(0.f_1f_2f_3\cdots f_m)_\beta \times \beta^{\pm E} \tag{1.1}$$

という形で表されている. ここで,

$$\pm\,(0.f_1f_2f_3\cdots f_m) = \pm(f_1\times\beta^{-1} + f_2\times\beta^{-2} + f_3\times\beta^{-3} + \cdots f_m\times\beta^{-m}) \tag{1.2}$$

は, 仮数部で f_iは 0 から$\beta-1$ までの整数, $f_1\neq 0$ である. また, Eは指数部で 0 または正の整数である.

ほとんどのメインフレーム系の計算機で使用されている実数表現は IBM 形式と呼ばれるものである. これに対して, UNIX などのコンパイラでは IEEE(アイ ・トリプル ・イーと読む) 規格を採用しているものが多い. どちらも, 32 ビットを 1 語とする点は同じであるが, 採用されている進法が異なり, 表現できる実数の範囲や精度に違いがある.

- **IBM 形式**

  これは富士通, 日立などのいわゆる IBM コンパチ汎用機で採用されている 16 進法の浮動小数点形式である. これらの計算機はもともと事務処理などのデータ処理が便利なように作られているため, 16 進演算が基本になっている. 各ビットの使い方は以下の通り.

0	1 2 3 4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31
±	$E+64$	f_1	f_2	f_3	f_4	f_5	f_6

  ここで, 指数部は負にならないように Eに 64 を足した 「ゲタばき表現」で表され, Eは-64 から 63 の値をとる. 仮数部は 16 進の 6 桁で, 7 桁目は切捨てられる. 0 は, この表示体系にはなじまない数値であるので, 別途, 「全てのビットが 0 である数値」という定義がなされている.

  この方法で表される 0 でない数値のうち絶対値最大のものは

$$(1-16^{-6})\times 16^{63} = 7.237005\times 10^{75} \tag{1.3}$$

  であり, 絶対値最小のものは

$$16^{-1}\times 16^{-64} = 5.397605\times 10^{-79} \tag{1.4}$$

  である.

  また, この方法で表される数値の相対誤差は $f_1=f_2=\cdots=f_6=15$ の時, 最も小さくて

$$\text{約}\ 16^{-6}\approx 6\times 10^{-8} \tag{1.5}$$

$f_1 = 1, f_2 = \cdots = f_6 = 0$ の時, 最も大きくて

$$約\ 16^{-5} \approx 10^{-6} \tag{1.6}$$

である. このように f_1の値が小さいと, その上位ビットが無駄になってしまうので精度が極端に悪くなる.

- **IEEE 形式**

  これは UNIX マシンなどで採用されている 2 進法の浮動小数点形式である. 最近では, この形式を採用する計算機が多い. これは, IBM 形式に比べて相対精度が高いという特徴がある. 各ビットの使い方は以下の通り.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
±	$E + 126$								$f_2 \cdot\cdot f_4$			$f_5 \cdots f_8$				$f_9 \cdots f_{12}$				$f_{13} \cdots f_{16}$				$f_{17} \cdots f_{20}$				$f_{21} \cdots f_{24}$			

  ここで, 指数部は Eに 126 を足した 「ゲタばき表現」で表され, Eは-125 から 128 の値をとる. 仮数部は 2 進の 24 桁で, f_1は常に 1 と決っているので省略される. ただし, Eが-126 (ゲタばき表現の指数部が 0) の時には, $f_1 = 0$ と見なして, より小さな数を表現するが, 当然 2 進 24 桁の精度はない. また, $E = 129$ (ゲタばき表現の指数部が 255) の時は, 無限大など特殊な数字を表すのに用いられ, 実数演算はできない. 0 は全てのビットが 0 である.

  この方法で表される正規数 $(f_1 \neq 0)$ のうち絶対値最大のものは

$$(1 - 2^{-24}) \times 2^{128} = 3.40282347 \times 10^{38} \tag{1.7}$$

  であり, 絶対値最小のものは

$$2^{-1} \times 2^{-125} = 1.17549435 \times 10^{-38} \tag{1.8}$$

  である. 非正規数 $(f_1 = 0)$ まで含めて, 絶対値最小のものは,

$$2^{-24} \times 2^{-125} = 1.40129846 \times 10^{-45} \tag{1.9}$$

  である.

  また, この方法で表される正規数の相対誤差は $f_1 = f_2 = \cdots = f_{24} = 1$ の時, 最も小さくて

$$約\ 2^{-25} \approx 3 \times 10^{-8} \tag{1.10}$$

  $f_1 = 1, f_2 = \cdots = f_6 = 0$ の時, 最も大きくて

$$約\ 2^{-24} \approx 6 \times 10^{-8} \tag{1.11}$$

  である. ただし, これは実数を丸める時に 25 桁目を 0 捨 1 入 (四捨五入の 2 進数版) した場合で, IEEE 規格ではその他の丸めも許されている. 切捨ての場合は, この 2 倍になる. もちろん, 非正規数はこの限りではない.

このように, 同じビット数で実数を表現しても, 実数として表現できる範囲や精度はシステムによって異なる. 電脳ライブラリでは, このようなシステム依存の定数を MATH1/SYSLIB の GLpSET/GLpGET で管理している.

我々の身の回りにあるコンピュータで採用されている表現方法を以下の表に示す.

計算機	OS	コンパイラ	浮動小数点表現方法	備考
FACOM	MSP(汎用)	FORT77EX	IBM	
FACOM	XMP(UNIX)	FORT77EX	IBM	
HITAC	VOS3(汎用)	??	IBM	
HITAC	HIUXM(UNIX)	f77	IBM	
SUN	UNIX	SUN FORTRAN	IEEE	
PC9801	MS−DOS	F77L(Lahey)	IEEE	
PC9801	MS−DOS	BASIC	その他*	

注 (*): 2 進表現で IEEE に近いが, 符号ビットの位置や指数部のゲタの値などが異なる.

1.4 文字の内部表現

文字の内部表現も実数表現と同様に機種に依存する. CHGLIB, CHKLIB は, 機種依存する文字処理を規格化するためのパッケージである.

FORTRAN では文法上, 以下の FORTRAN 文字集合が定められており, FORTRAN プログラムは, これらの文字だけで書かなければならない. (注釈行や文字型データの中は例外)

英字:	ABCDEFGHIJKLMNOPQRSTUVWXYZ
数字:	0123456789
特殊文字:	空白'()*+,-./:= 通貨記号

したがって, 小文字で FORTRAN プログラムを書くのは厳密に言えば文法違反になる.

これらの FORTRAN 文字を含めて, 文字の内部表現方法は FORTRAN の規格では特に規定されていない. 実数の内部表現方法と同様に, IBM 規格である EBCDIC と, アメリカの標準規格である ASCII とにわかれる.

- **EBCDIC**

  IBM によって定められた拡張 2 進化 10 進情報交換用コード (Extended Binary Coded Decimal Interchange Code). その名が示す通り, BCD(2 進化 10 進) コードと呼ばれるものを拡張したものである. BCD コードとは 16 進数の A,B,C,D,E,F を使わず, 0 から 9 までの数字を使って 10 進数を表現するもので, 4 ビットで 10 進数の 1 桁に対応させる.

  EBCDIC では 8 ビットで 1 文字を表現し, 数字は上位 4 ビットを F として下位 4 ビットを BCD コードに対応させている. (これが EBCDIC たる由縁) 富士通, 日立などのいわゆる IBM コンパチ汎用機でも採用されているが, 各社, 微妙に定義が異なる.

  FORTRAN 文字に関しては各社とも同じコードを使っているようであるが, その他の文字についてはかなり機種依存性が強い. 特に, 日本ではカタカナを使うためにコード体系を変更している場合があり, その対応の仕方に 2 種類ある. 一つは, アルファベットの小文字の部分をカタカナに置き換えてしまう方法で, この方法では, カタカナとアルファベットの小文字は同時に使用できない. 富士通の M シリーズではこの方法が採用されている (計算センターによって異なる場合がある). もう一つは, 小文字の部分を上記のようにカタカナに置き換えた上で, さらに, 小文字をあいているコードに押し込めたもので, EBCDIK(最後の K はカナ) コードとも呼ばれる. これは日立の M シリーズで採用されている. これら 2 つの方法で, アルファベットの大文字とカタカナに関しては同一のコードとなるが, 小文字に関しては互換性がなくなる.

- **ASCII コード**

  これはアメリカ規格協会 (ANSI) で規定された文字コード体系である. UNIX, MS-DOS などで採用されている. 日本では, ほぼ同じものが JIS X0201 として規定されている. ASCII コードは 7 ビットで, 最上位桁は 0 であるが, JIS には 8 ビット全部使ってカタカナまで規定した 8 単位符号表がある.

このように文字コードは, たとえ FORTRAN 文字であっても機種に依存する. したがって, 文字コードを返す `ICHAR` 関数は FORTRAN の標準の組み込み関数であるが, その値は機種に依存することになる. 文字

コードの大小を機種に依存せずに比較するための関数として, LGE, LGT, LLE, LLT があり, これはその計算機のコード体系によらず, ASCII コード順に比較する.

EBCDIC コード表

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

- 制御コードの意味に関しては, この節の最後の表を参照のこと.
- 富士通のコードにはこれ以外の制御コードも定義されている.
- 日立の EBCDIK ではアルファベットの小文字のコードが異なる.
- 特殊文字部分の※ 印は富士通と日立で定義が異なる部分である.
- ¥(5B) は通貨記号であり, アメリカでは $ となる. その場合には, E0 の$は\ (バックスラッシュ) となる.

ASCII コード表

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

JIS X0201 では上の表の\が¥, ~が‾(上線) となっている.

制御コード表

記号	意味	英語名	記号	意味	英語名
NUL	空値	null	DC1	装置制御 1	device control 1
SOH	ヘディング開始	start of heading	DC2	装置制御 2	device control 2
STX	テキスト開始	start of text	DC3	装置制御 3	device control 3
ETX	テキスト終了	end of text	DC4	装置制御 4	device control 4
EOT	伝送終了	end of transmission	NAK	否定応答	negative acknowledge
ENQ	問い合わせ	enquiry	SYN	同期信号	synchronous idle
ACK	肯定応答	acknowledge	ETB	伝送ブロック終結	end of transmission block
BEL	ベル	bell	CAN	取消	cancel
BS	後退	backspace	EM	媒体終端	end of medium
HT	水平タブ	horizontal tabulation	SUB	置換文字	substitute character
LF	改行	line feed	ESC	拡張	escape
VT	垂直タブ	horizontal tabulation	FS	ファイル分離文字	file separator
FF	改ページ	form feed	GS	グループ分離文字	group separator
CR	復帰	carriage return	RS	レコード分離文字	record separator
SO	シフトアウト	shift out	US	ユニット分離文字	unit separator
SI	シフトイン	shift in	SP	空白	space
DLE	伝送制御拡張	data link escape	DEL	消去	delete

富士通と日立の EBCDIC と ASCII の対応

コード	富士通	日立	ASCII
4A	c*	[	[
4F	|	!	!
5A	!	]	]
5B	¥	¥	$
5F	¬	^	^
6A		|	|
E0	$	$	\

- 注 1:富士通の 4A は c に縦棒.
- 注 2:ASCII コードとの対応は, 通常アスキー端末に出力されるコードで, これ以外のコードに変換される場合もある.

1.5 ファイルの構造

ファイルの形式も機種によりかなり異なるが, FIOLIB は固定長レコードのファイルに限って異機種間のデータ転送を可能にするためのユーティリティーである.

FORTRAN 規格におけるファイルの属性は基本的に, 書式のありなし (FORMATTED/UNFORMATTED) とアクセス方法 (SEQUENTIAL/DIRECT) の組み合せで決る. 書式のありなしは, 例えば WRITE 文を実行するときに, 内部表現を文字に変換するか, 内部表現をそのまま出力するかということを指定するものであり, いわば 「ファイルの中身」を指定するものである. 一方, アクセス方法の指定は文字どおり解釈すれば, 順番に読み書きするか, ランダムに読み書きするかという指定であるが, 実際には 「アクセス方法」よりも, そのアクセス方法を実現する 「ファイルの構造」の方が問題となる場合が多い.

規格上, DIRECT ファイルではレコード長を指定する RECL 指定子を書かなければならず, SEQUENTIAL ファイルの場合には, これを書いてはいけない. つまり, 「アクセス方法」の指定は, DIRECT ファイルは固定長レコードのファイルであり, SEQUENTIAL ファイルは可変長レコードのファイルであるという「ファイル構造」を暗黙のうちに指定することになる.

FORTRAN プログラム上の論理的なレコードが, 実際に記録媒体の上でどのように記録されるかということに関しても, 大きく分けてメインフレーム系 (IBM 系) と UNIX 系の 2 種類あり, それぞれファイルの扱い方がかなり異なる.

1.5.1 メインフレーム系

メインフレーム系の計算機では, アクセス方法とファイルの構造は, ほとんど独立した概念として扱われる. ファイルの構造としては, 固定長 (F), 可変長 (V), ブロック化固定長 (FB), ブロック化可変長 (VB), さらに, スパンドブロック化可変長 (VBS) などがあるが, これらの概念は FORTRAN にはないので, FORTRAN プログラム以外の部分 (DD 文など) で指定することになる.

これらのファイル構造のうち, 固定長ファイル (F, FB) が最も単純で, 決った長さのデータが並んでいるだけで, 余分なコードは入っていない. これに対して, スパンドブロック化可変長 (VBS) 形式は, 最も複雑な構造をしているが, どんな長さのレコードも記録できる真の可変長レコード形式である (V, VB は可変長といってもレコード長の上限がある).

また, 一般に計算機が読み書きする時には, できるだけ大きな単位にまとめて行う方が効率がよいので, ブロック化された形式の方が処理は早い.

- **SEQUENTIAL ファイル**

  書式なしの SEQUENTIAL ファイルでは, 通常 VBS 形式が使われる. この形式が最も FORTRAN 規格とは整合性がよく, DD 文などで何も指定しなければ, この形式が採用される場合が多い.

  書式つきの場合は, もともと, 特定の長さを持つデバイスに出力することが多いので, F, FB なども使われる. この場合, プログラムが書き出すレコードの最大レコード長が, ファイルのレコード長を越えることがないように注意しなければならない.

  書式なしのデータを F, FB などに出力することは, FORTRAN の文法との整合性が悪いので, 禁止されているか, 制御コードを含む形式のファイルとして扱われることもある.

どちらにしても, ユーザーがブロック長などを指定するということは, 単なる FORTRAN プログラマの知識を越えて, システム管理者的な知識が必要となり面倒である.

しかし逆に言えば, プログラムで行う処理の性質とシステムの特性の両方を考慮して極限までチューンアップすることが可能である. 実際, メインフレーム系の計算機では, ブロック長などのパラメータの選び方でかなり処理速度に差が出る.

- **DIRECT ファイル**

  DIRECT ファイルで使えるのは固定長レコードのファイルだけである. DIRECT ファイルはランダムにアクセスすることが可能なので, ブロック化ができない. したがって, 順番にアクセスするならば SEQUENTIAL の方が早くなる.

  一般的に, メインフレーム系の計算機では DIRECT ファイルは扱いやすくないので, あまり使われていないようである.

1.5.2 UNIX 系

UNIX 系の計算機が扱うファイルは基本的に全て可変長レコードのファイルであり, 表向きブロック化という概念はない. 実際には, 何等かのブロック化が計算機内部で行われているものと思われるが, それは, FORTRAN ユーザーのレベルではわからない.

- **SEQUENTIAL ファイル**

  書式つきの SEQUENTIAL ファイルでは通常のテキストファイルと同様にレコードの終りに行末記号が付加される. 行末記号は UNIX では 1 バイトで LF, MS-DOS では 2 バイトで LF, CR である.

  書式なしの場合には, レコードの最初と終り (または最初だけ) に, レコードを認識する何等かの記号が付加される. この記号は処理系によってかなりことなり, 同じ内部コードを採用している場合でも互換性はない.

- **DIRECT ファイル**

  DIRECT ファイルではレコードの長さが決っているので, レコードを認識するための記号などは付加されず, 決った長さのレコードがつながった最も単純なファイルとなる. (システムによってはファイルの先頭に固有のヘッダを付加するものもある.)

  したがって, 変数の内部表現が同じであれば, ほとんどの計算機で互換性がとれる. さらに, システムによっては SEQUENTIAL ファイルに比べて, 処理速度が圧倒的に早い場合もある.

  UNIX 系の計算機においては, たとえ単に順番にレコードを参照する場合でも, 利用価値のあるファイル形式である.

第 2 章

BITLIB : ビットパターンの処理 *☆

2.1 概要

ビットパターンの処理をおこなう関数 ･サブルーチンパッケージ. 1 語長が 32bit のシステムを念頭においている.

ビットパターンを与える/返す引数は 1 語長であれば, 整数型である必要はない.

なおこのパッケージは (BITPIC, BITPCI を除いて), NCAR のグラフィックパッケージに付随していた基本関数 ･サブルーチン群にもとづいている.

2.2 サブルーチンのリスト

BITPIC(IP,CP)	ビットパターンを文字列化する.
BITPCI(CP,IP)	文字列をビットパターン化する.
GBYTE(NP,IO,IB,NB)	ビットパターンを切り出す.
GBYTES(NP,NO,IB,NB,NS,IT)	ビットパターンを切り出す.
SBYTE(NP,IO,IB,NB)	ビットパターンを埋め込む.
SBYTES(NP,NO,IB,NB,NS,IT)	ビットパターンを埋め込む.

2.3 関数のリスト

ISHIFT(IW,N)	ビットパターンをシフトする.
IAND(I1,I2)	ビット積を求める.
IOR(I1,I2)	ビット和を求める.

IAND, IOR は実時間 FORTRAN に規定されており, 処理系が用意している場合がある.

2.4 サブルーチンの説明

2.4.1 BITPIC

1. 機能

   ビットパターンを'0', '1' の文字列として返す.

2. 呼び出し方法

```
CALL BITPIC(IP,CP)
```

3. パラメーターの説明

IP	(I)	ビットパターンを調べる 1 語長の引数.
CP	(C*(*))	ビットパターンを返す文字型の引数. ビットが ON なら'1', OFF なら'0' を返す.

4. 備考

(a) CP の長さが N (=LEN(CP)) ならば IP の下位 MIN(N,NB) ビット (NB は内部変数'NBITSPW' で決まる数) を調べそのパターンを右詰めして返す. N が NB より大きいとき CP(1:N-NB) は不定である.

2.4.2 BITPCI

1. 機能

'0', '1' の文字列をビットパターンとして返す.

2. 呼び出し方法

```
CALL BITPCI(CP,IP)
```

3. パラメーターの説明

CP	(C*(*))	ビットパターンを与える文字型の引数. ビットが ON なら'1', OFF なら'0' を指定する.
IP	(I)	ビットパターンを返す 1 語長の引数.

4. 備考

(a) CP の長さが N (=LEN(CP)) ならば CP の下位 MIN(N,NB) 文字 (NB は内部変数'NBITSPW' で決まる数) を調べそのパターンを右詰めして返す. N が NB より大きいとき, 下位 NB 文字 (CP(N-NB+1:N)) のパターンを調べる. ビットパターンの指定方法について, 正確には'0' ならビットが OFF, '0' 以外ならビットが ON と見なす.

2.4.3 GBYTE/GBYTES

1. 機能

連続した記憶領域からビットパターンを切り出す.

2. 呼び出し方法

```
CALL GBYTE(NP,IO,IB,NB)
CALL GBYTES(NP,NO,IB,NB,NS,IT)
```

3. パラメーターの説明

NP (I) 連続した記憶領域 (たとえば配列) の先頭となる語.
IO (I) 切り出したビットパターンを返す変数.
NO (I) 切り出したビットパターンを返す配列.
IB (I) 先頭の語においてスキップするビット数.
NB (I) 切り出すビット数.
NS (I) 切り出すビットパターン間のビット数.
IT (I) 配列 NO の長さ.

4. 備考

(a) NB は 1 以上, 1 語のビット数 (ここでは 32) 以下でなければならない.

2.4.4 SBYTE/SBYTES

1. 機能

連続した記憶領域にビットパターンを埋め込む.

2. 呼び出し方法

```
CALL SBYTE(NP,IO,IB,NB)
CALL SBYTES(NP,NO,IB,NB,NS,IT)
```

3. パラメーターの説明

NP (I) 連続した記憶領域 (たとえば配列) の先頭となる語.
IO (I) 埋め込むビットパターンを与える変数.
NO (I) 埋め込むビットパターンを与える配列.
IB (I) 先頭の語においてスキップするビット数.
NB (I) 埋め込むビット数.
NS (I) 埋め込むビットパターン間のビット数.
IT (I) 配列 NO の長さ.

4. 備考

(a) NB は 1 以上, 1 語のビット数 (ここでは 32) 以下でなければならない.

2.5 関数の説明

2.5.1 ISHIFT

1. 機能

ビットパターンをシフトする.

2. 呼び出し方法

```
ISHIFT(IW,N)
```

3. パラメーターの説明

IW	(I)	ビットパターンをシフトする 1 語長の引数.
N	(I)	シフトするビット数. N >0 なら左側にシフトする. 左側にあふれた分は右側に詰められる (circular shift). N <0 なら右側にシフトする. 右側にあふれた分は捨てられる (end-off shift).
ISHIFT	(I)	シフトした結果を返す関数値.

4. 備考

   (a) なし.

2.5.2 IAND/IOR

1. 機能

   ビットごとのブール積/和を求める.

2. 呼び出し方法

```
IAND(I1,I2)
IOR(I1,I2)
```

3. パラメーターの説明

I1, I2	(I)	ビット列を与える 1 語長の引数.
IAND	(I)	ビット列のブール積を返す関数値.
IOR	(I)	ビット列のブール和を返す関数値.

4. 備考

   (a) IAND, IOR は実時間 FORTRAN に規定されており, 処理系が用意している場合がある.

第 3 章

CHGLIB : 大文字 ・小文字の変換 *☆

3.1 概要

大文字と小文字の変換をおこなうサブルーチンパッケージ.

3.2 サブルーチンのリスト

CUPPER(CH) 文字列を大文字化する.
CLOWER(CH) 文字列を小文字化する.

3.3 サブルーチンの説明

3.3.1 CUPPER/CLOWER

1. 機能

   文字列を大文字化/小文字化する.

2. 呼び出し方法

```
CALL CUPPER(CH)
CALL CLOWER(CH)
```

3. パラメーターの説明

   CH (C*(*)) 処理する文字列. 入力パラメータでもあり出力パラメータでもある.

4. 備考

   (a) なし.

第 4 章

CHKLIB : 文字種の判別 *

4.1 概要

文字の種類を判別する論理型の関数パッケージ. このパッケージが判別する文字の種類には次のものがある.

- 空白 : ブランク (以下では △ であらわす).
- 通貨記号 : 処理系に依存するがたとえば$ (以下の説明では$を用いる).
- 特殊文字 : 次の 13 文字からなる. △, ', (,), *, +, ,, -, ., /, :, =, $.
- 英字 : 次の 26 文字からなる. A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z.
- 数字 : 次の 10 文字からなる. 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- 英数字 : 英字 ・数字からなる.
- FORTRAN 文字 : 英字 ・数字 ・特殊文字からなる.

4.2 関数のリスト

(C は長さ 1 の文字型の引数)

LCHRB(C)	空白かどうかを判別する.
LCHRC(C)	通貨記号かどうかを判別する.
LCHRS(C)	特殊文字かどうかを判別する.
LCHRL(C)	英字かどうかを判別する.
LCHRD(C)	数字かどうかを判別する.
LCHRA(C)	英数字かどうかを判別する.
LCHRF(C)	FORTRAN 文字かどうかを判別する.
LCHR(CHAR,CREF)	CHAR が CREF で指定した文字種かどうかを判別する. CHAR と CREF の長さは等しくなければならない. CREF で指定できるのは 'B' : 空白, 'C' : 通貨記号, 'S' : 特殊文字, 'L' : 英字, 'D' : 数字, 'A' : 英数字, 'F' : FORTRAN 文字を組み合わせたものである.

4.3 関数の説明

4.3.1 LCHRB/LCHRC/LCHRS/LCHRL/LCHRD/LCHRA/LCHRF

1. 機能

   文字の種類を判別する.

2. 呼び出し方法

```
LCHRB(C)
LCHRC(C)
LCHRS(C)
LCHRL(C)
LCHRD(C)
LCHRA(C)
LCHRF(C)
```

3. パラメーターの説明

C	(C*1)	文字種類を調べる長さ 1 の文字型の引数.
LCHRB	(L)	C が空白なら .TRUE., そうでなければ.FALSE. を返す論理型関数値.
LCHRC	(L)	C が通貨記号なら .TRUE., そうでなければ.FALSE. を返す論理型関数値.
LCHRS	(L)	C が特殊文字なら .TRUE., そうでなければ.FALSE. を返す論理型関数値.
LCHRL	(L)	C が英字なら .TRUE., そうでなければ.FALSE. を返す論理型関数値.
LCHRD	(L)	C が数字なら .TRUE., そうでなければ.FALSE. を返す論理型関数値.
LCHRA	(L)	C が英数字なら .TRUE., そうでなければ.FALSE. を返す論理型関数値.
LCHRF	(L)	C が FORTRAN 文字なら .TRUE., そうでなければ.FALSE. を返す論理型関数値.

4. 備考

   (a) なし

4.3.2 LCHR

1. 機能

   文字列の種類を判別する. 調べる文字列の種類は, テンプレートとして別の文字列で与える.

2. 呼び出し方法

```
LCHR(CHAR,CREF)
```

3. パラメーターの説明

CHAR	(C*(*))	文字種類を調べる文字列.
CREF	(C*(*))	文字列の種類を与えるテンプレート文字列. 'B' : 空白, 'C' : 通貨記号, 'S' : 特殊文字, 'L' : 英字, 'D' : 数字, 'A' : 英数字, 'F' : FORTRAN 文字を組み合わせたもので表現する.
LCHR	(L)	CHAR が CREF で指定した文字種と一致していれば .TRUE., そうでなければ .FALSE. を返す論理型関数値.

4. 備考

(a) CHAR と CREF の長さは等しくなければならない.

第 5 章

CHNLIB : 文字列の置換 *

5.1 概要

文字列の指定された部分に文字列 ・整数 ・実数を書き込むサブルーチンパッケージ.

5.2 サブルーチンのリスト

CHNGC(CH,CA,CB)	文字列 CH の中で文字列 CA と一致する部分を文字列 CB でおきかえる.
CHNGI(CH,CA,II,CFMT)	文字列 CH の中で文字列 CA と一致する部分を書式仕様 CFMT にしたがって整数 II でおきかえる.
CHNGR(CH,CA,RR,CFMT)	文字列 CH の中で文字列 CA と一致する部分を書式仕様 CFMT にしたがって実数 RR でおきかえる.

5.3 サブルーチンの説明

5.3.1 CHNGC

1. 機能

   あたえられた文字列の中から指定された文字列を見つけだし, その部分を別の文字列でおきかえる.

2. 呼び出し方法

   ```
   CALL CHNGC(CH,CA,CB)
   ```

3. パラメーターの説明

CH	(C*(*))	調べる文字列.
CA	(C*(*))	見つけ出す文字列.
CB	(C*(*))	おきかえる文字列.

4. 備考

   (a) CA と CB の長さは同じでなければならない.

(b) CH の長さは CA の長さ以上でなければならない.

(c) CA が CH の中に見つからなかっときは警告メッセージが出力される.

5.3.2 CHNGI/CHNGR

1. 機能

   あたえられた文字列の中から指定された文字列を見つけだし, その部分を書式にしたがって整数/実数でおきかえる.

2. 呼び出し方法

```
CALL CHNGI(CH,CA,II,CFMT)
CALL CHNGR(CH,CA,RR,CFMT)
```

3. パラメーターの説明

CH	(C*(*))	調べる文字列.
CA	(C*(*))	見つけ出す文字列.
II	(I)	おきかえる整数値.
RR	(R)	おきかえる実数値.
CFMT	(C*(*))	文字書式仕様. たとえば, '(I4)', '(E12.4)' と指定する.

4. 備考

   (a) CH の長さは CA の長さ以上でなければならない.

   (b) CA が CH の中に見つからなかっときは警告メッセージが出力される.

   (c) 書式仕様の整合性はチェックされない.

第 6 章

FMTLIB : 数値の文字列化 *

6.1 概要

数値を編集して文字列化するサブルーチンパッケージ.

6.2 サブルーチンのリスト

CHVAL(CFMT,VAL,CVAL) 数値を文字列化する

6.3 サブルーチンの説明

6.3.1 CHVAL

1. 機能

   実数値を文字書式仕様にしたがって文字列化する. 8 文字以内におさまるよう (有効数字は 3 桁以下) 自動的に書式仕様を生成するオプションもある.

2. 呼び出し方法

```
CALL CHVAL(CFMT,VAL,CVAL)
```

3. パラメーターの説明

CFMT	(C*(*))	文字書式仕様あるいは CHVAL が解釈するオプション. CFMT の最初の文字が'(' で始まるとき文字書式仕様と解釈する. これ以外のときのオプションとして次の 4 つが指定可能である. 'A': 書式仕様を自動的に生成する. 'B': 'A' かつ後続の'0' と小数点を除く. 'C': 'B' かつ小数点の前の'0' を除く. 'D': 'C', ただし指数形式のときに限る. たとえば CFMT として'A', '(F6.1)' のように指定する.
VAL	(R)	文字化する実数値.
CVAL	(C*(*))	文字化された実数値を返す長さが 8 文字以上の文字型の引数.

4. 備考

(a) CFMT の長さは 16 文字以下でなければならない.

(b) CFMT として整数型の編集記述子も指定できる. このとき VAL の小数点以下を四捨五入して文字列化される.

(c) CHVAL は自動的に文字を左詰めする. つまり実数値 1.2 を文字化するために'(F4.1)' としても '(F8.1)' としても CHVAL が返す値はどちらも'1.2' である.

(d) システムによっては, F 型編集記述子によって 1 より小さい値を編集するとき, 小数点の左隣にゼロがつかない場合があるが, このルーチンはそのような場合'0' (ゼロ) を補う.

(e) 指数部を示す文字は'e' (小文字) を用いている.

第 7 章

DATELIB : 日付の取り扱い *☆

7.1 概要

日付を扱う関数・サブルーチンパッケージ. このパッケージの説明では, 次の用語を用いる.

- 1 型の日付 (IDATE) – 上 5 桁以上が年, 3,4 桁が月, 1,2 桁が日の 1 整数で日付をあらわす. 例: 19920401.
- 2 型の日付 (IY,ITD) – 年と通しの日数の 2 整数で日付をあらわす. 例: 1992, 92.
- 3 型の日付 (IY,IM,ID) – 年, 月, 日の 3 整数で日付を表す. 例: 1992, 4, 1.
- 曜日番号 – 1:日曜, 2:月曜, 3:火曜, 4:水曜, 5:木曜, 6:金曜, 7:土曜 であるような整数値.

7.2 サブルーチンのリスト

DATE12(IDATE,IY,ITD)	1 型の日付を 2 型の日付に変換する.
DATE13(IDATE,IY,IM,ID)	1 型の日付を 3 型の日付に変換する.
DATE21(IDATE,IY,ITD)	2 型の日付を 1 型の日付に変換する.
DATE23(IY,IM,ID,ITD)	2 型の日付を 3 型の日付に変換する.
DATE31(IDATE,IY,IM,ID)	3 型の日付を 1 型の日付に変換する.
DATE32(IY,IM,ID,ITD)	3 型の日付を 2 型の日付に変換する.
DATEF1(N,IDATE,NDATE)	IDATE の N 日後 (NDATE) を求める.
DATEF2(N,IY,ITD,NY,NTD)	IY,ITD の N 日後 (NY,NTD) を求める.
DATEF3(N,IY,IM,ID,NY,NM,ND)	IY,IM,ID の N 日後 (NY,NM,ND) を求める.
DATEG1(N,IDATE,NDATE)	IDATE の何 (N) 日後が NDATE かを求める.
DATEG2(N,IY,ITD,NY,NTD)	IY,ITD の何 (N) 日後が NY,NTD かを求める.
DATEG3(N,IY,IM,ID,NY,NM,ND)	IY,IM,ID の何 (N) 日後が NY,NM,ND かを求める.
DATEQ1(IDATE)	今日の 1 型の日付を求める.
DATEQ2(IY,ITD)	今日の 2 型の日付を求める.
DATEQ3(IY,IM,ID)	今日の 3 型の日付を求める.
DATEC1(CFORM,IDATE)	IDATE を CFORM に従って表現して CFORM で返す.
DATEC2(CFORM,IY,ITD)	IY,ITD を CFORM に従って表現して CFORM で返す.
DATEC3(CFORM,IY,IM,ID)	IY,IM,ID を CFORM に従って表現して CFORM で返す.

CFORM 中で使えるキーとなる文字は, 'Y' : 年, 'M' : 月, 'D' : 日, 'C' : 文字型の月, 'W' : 曜日, である. たとえば IDATE=19920401 (この日は水曜日) に対して CFORM='CCC,DD,YY (WWW)' と指定して DATEC1 を呼ぶと CFORM='APR, 1,92 (WED)' が返される.

7.3 関数のリスト

NDATE1(IDATE,NDATE)	DATEG1(NDATE1,IDATE,NDATE) と同じ.
NDATE2(IY,ITD,NY,NTD)	DATEG2(NDATE2,IY,ITD,NY,NTD) と同じ.
NDATE3(IY,IM,ID,NY,NM,ND)	DATEG3(NDATE3,IY,IM,ID,NY,NM,ND) と同じ.
IWEEK1(IDATE)	1 型の日付 IDATE に対応する曜日番号を返す.
IWEEK2(IY,ITD)	2 型の日付 IY,ITD に対応する曜日番号を返す.
IWEEK3(IY,IM,ID)	3 型の日付 IY,IM,ID に対応する曜日番号を返す.
NDMON(IY,IM)	IY 年 IM 月は何日あるかを返す.
NDYEAR(IY)	IY 年は何日あるかを返す.
CMON(IM)	文字型の月名を返す文字型関数. 文字の長さはユーザーが指定すること.
CWEEK(IW)	文字型の曜日を返す文字型関数. 文字の長さはユーザーが指定すること.

7.4 サブルーチンの説明

7.4.1 DATE12/DATE13/DATE21/DATE23/DATE31/DATE32

1. 機能

   日付の型を変換する.
   DATE12 : 1 型の日付を 2 型の日付に変換する.
   DATE13 : 1 型の日付を 3 型の日付に変換する.
   DATE21 : 2 型の日付を 1 型の日付に変換する.
   DATE23 : 2 型の日付を 3 型の日付に変換する.
   DATE31 : 3 型の日付を 1 型の日付に変換する.
   DATE32 : 3 型の日付を 2 型の日付に変換する.

2. 呼び出し方法

```
CALL DATE12(IDATE,IY,ITD)
CALL DATE13(IDATE,IY,IM,ID)
CALL DATE21(IDATE,IY,ITD)
CALL DATE23(IY,IM,ID,ITD)
CALL DATE31(IDATE,IY,IM,ID)
CALL DATE32(IY,IM,ID,ITD)
```

3. パラメーターの説明

IDATE	(I)	1 型の日付.
IY	(I)	年.
IM	(I)	月.
ID	(I)	日.
ITD	(I)	通しの日付.

4. 備考

   (a) なし.

7.4.2 DATEF1/DATEF2/DATEF3

1. 機能

   N 日後の日付を求める.
   DATEF1 : IDATE の N 日後 (NDATE) を求める.
   DATEF2 : IY,ITD の N 日後 (NY,NTD) を求める.
   DATEF3 : IY,IM,ID の N 日後 (NY,NM,ND) を求める.

2. 呼び出し方法

```
CALL DATEF1(N,IDATE,NDATE)
CALL DATEF2(N,IY,ITD,NY,NTD)
CALL DATEF3(N,IY,IM,ID,NY,NM,ND)
```

3. パラメーターの説明

IDATE, NDATE	(I)	1 型の日付.
IY, NY	(I)	年.
IM, NM	(I)	月.
ID, ND	(I)	日.
ITD, NTD	(I)	通しの日付.
N	(I)	日付の差 (日数).

4. 備考

   (a) N としては負の値を指定してもよい. つまり負の N に対して−N 日前の日付も調べることができる.

7.4.3 DATEG1/DATEG2/DATEG3

1. 機能

   日付の差を求める.
   DATEG1 : IDATE の何 (N) 日後が NDATE かを求める.
   DATEG2 : IY,ITD の何 (N) 日後が NY,NTD かを求める.
   DATEG3 : IY,IM,ID の何 (N) 日後が NY,NM,ND かを求める.

2. 呼び出し方法

```
CALL DATEG1(N,IDATE,NDATE)
CALL DATEG2(N,IY,ITD,NY,NTD)
CALL DATEG3(N,IY,IM,ID,NY,NM,ND)
```

3. パラメーターの説明

IDATE, NDATE	(I)	1 型の日付.
IY, NY	(I)	年.
IM, NM	(I)	月.
ID, ND	(I)	日.
ITD, NTD	(I)	通しの日付.
N	(I)	日付の差 (日数).

4. 備考

(a) 過去にさかのぼって日付の差を求めることもできる. つまり日付の差として N は負にもなりうる.

7.4.4 DATEQ1/DATEQ2/DATEQ3

1. 機能

今日の日付を求める.
DATEQ1 : 今日の 1 型の日付を求める.
DATEQ2 : 今日の 2 型の日付を求める.
DATEQ3 : 今日の 3 型の日付を求める.

2. 呼び出し方法

```
DATEQ1(IDATE)
DATEQ2(IY,ITD)
DATEQ3(IY,IM,ID)
```

3. パラメーターの説明

IDATE	(I)	1 型の日付.
IY	(I)	年.
IM	(I)	月.
ID	(I)	日.
ITD	(I)	通しの日付.

4. 備考

(a) これらはシステムに依存したサブルーチンなので, 正しく移植されていない可能性もある.

7.4.5 DATEC1/DATEC2/DATEC3

1. 機能

フォーマットを指定して日付を文字列で表現する.
DATEC1 : 1 型の日付を文字列で表現する.
DATEC2 : 2 型の日付を文字列で表現する.
DATEC3 : 3 型の日付を文字列で表現する.

2. 呼び出し方法

```
CALL DATEC1(CFORM,IDATE)
CALL DATEC2(CFORM,IY,ITD)
CALL DATEC3(CFORM,IY,IM,ID)
```

3. パラメーターの説明

IDATE	(I)	1 型の日付.
IY	(I)	年.
IM	(I)	月.
ID	(I)	日.
ITD	(I)	通しの日付.
CFORM	(C*(*))	日付のフォーマット. 入力パラメータでもあり, 出力パラメーターでもある. CFORM 中で使えるキーとなる文字は, 'Y' : 年, 'M' : 月, 'D' : 日, 'C' : 文字型の月, 'W' : 曜日, である. たとえば IDATE=19920401 (この日は水曜日) に対して CFORM='CCC,DD,YY (WWW)' と指定して DATEC1 を呼ぶとCFORM='APR,␣1,92␣(WED)' が返される.

4. 備考

(a) 日付のフォーマットの指定方法について, 正確には, キーとなる文字が最初に現れたところから最後に現れたところの間の部分をおきかえる. したがって例えば, 'CCCCC/YYYY' と書くのと 'CAAAC/YBBY と書くのは同じである.

(b) 文字型の月名は右寄せして出力される.

7.5 関数の説明

7.5.1 NDATE1/NDATE2/NDATE3

1. 機能

日付の差を求める.
NDATE1 : IDATE の何日後が NDATE かを求める.
NDATE2 : IY,ITD の何日後が NY,NTD かを求める.
NDATE3 : IY,IM,ID の何日後が NY,NM,ND かを求める.

2. 呼び出し方法

```
NDATE1(IDATE,NDATE)
NDATE2(IY,ITD,NY,NTD)
NDATE3(IY,IM,ID,NY,NM,ND)
```

3. パラメーターの説明

IDATE, NDATE	(I)	1 型の日付.
IY, NY	(I)	年.
IM, NM	(I)	月.
ID, ND	(I)	日.
ITD, NTD	(I)	通しの日付.
NDATE1	(I)	1 型の日付の差を与える関数値.
NDATE2	(I)	2 型の日付の差を与える関数値.
NDATE3	(I)	3 型の日付の差を与える関数値.

4. 備考

(a) なし

7.5.2 IWEEK1/IWEEK2/IWEEK3

1. 機能

曜日番号を求める.
IWEEK1：1 型の日付に対応する曜日番号を求める.
IWEEK2：2 型の日付に対応する曜日番号を求める.
IWEEK3：3 型の日付に対応する曜日番号を求める.

2. 呼び出し方法

```
IWEEK1(IDATE)
IWEEK2(IY,ITD)
IWEEK3(IY,IM,ID)
```

3. パラメーターの説明

IDATE	(I)	1 型の日付.
IY	(I)	年.
IM	(I)	月.
ID	(I)	日.
ITD	(I)	通しの日付.
IWEEK1	(I)	1 型の日付に対応する曜日番号を与える関数値.
IWEEK2	(I)	2 型の日付に対応する曜日番号を与える関数値.
IWEEK3	(I)	3 型の日付に対応する曜日番号を与える関数値.

4. 備考

(a) なし

7.5.3 NDMON/NDYEAR

1. 機能

NDMON：IY 年 IM 月は何日あるかを返す.
NDYEAR：IY 年は何日あるかを返す.

2. 呼び出し方法

```
NDMON(IY,IM)
NDYEAR(IY)
```

3. パラメーターの説明

IY	(I)	年.
IM	(I)	月.
NDMON	(I)	指定した月の日数を与える関数値.
NDYEAR	(I)	指定した年の日数を与える関数値.

4. 備考

(a) なし.

7.5.4 CMON

1. 機能

文字型の月名を返す文字型関数.

2. 呼び出し方法

```
CMON(IM)
```

3. パラメーターの説明

IM	(I)	月.
CMON	(C*(*))	文字型の月名を返す文字型関数値. 文字の長さはユーザーが指定すること.

4. 備考

(a) 文字型の月名とは, JANUARY, FEBRUARY...のことである. 先頭から LEN(CMON) の長さの分だけ返される.

7.5.5 CWEEK

1. 機能

文字型の曜日を返す文字型関数.

2. 呼び出し方法

```
CWEEK(IW)
```

3. パラメーターの説明

IW	(I)	曜日番号.
CWEEK	(C*(*))	文字型の曜日を返す文字型関数値. 文字の長さはユーザーが指定すること.

4. 備考

(a) 文字型の曜日とは, SUNDAY, MONDAY... のことである. 先頭から LEN(CWEEK) の長さの分だけ返される.

第 8 章

TIMELIB : 時刻の取り扱い *☆

8.1 概要

時刻を扱うサブルーチンパッケージ. このパッケージの説明では, 次の用語を用いる.

- 1 型の時刻 (ITIME) – 上 5, 6 桁が時, 3,4 桁が分, 1,2 桁が秒の 1 整数で時刻をあらわす. 例: 100930.
- 2 型の時刻 (ITT) – 通しの秒数の 1 整数で時刻をあらわす. 例: 36570.
- 3 型の時刻 (IH,IM,IS) – 時, 分, 秒の 3 整数で時刻を表す. 例: 10, 9, 30.

8.2 サブルーチンのリスト

TIME12(ITIME,ITT)	1 型の時刻を 2 型の時刻に変換する.
TIME13(ITIME,IH,IM,IS)	1 型の時刻を 3 型の時刻に変換する.
TIME21(ITIME,ITT)	2 型の時刻を 1 型の時刻に変換する.
TIME23(IH,IM,IS,ITT)	2 型の時刻を 3 型の時刻に変換する.
TIME31(ITIME,IH,IM,IS)	3 型の時刻を 1 型の時刻に変換する.
TIME32(IH,IM,IS,ITT)	3 型の時刻を 2 型の時刻に変換する.
TIMEQ1(ITIME)	現在の 1 型の時刻を求める.
TIMEQ2(ITT)	現在の 2 型の時刻を求める.
TIMEQ3(IH,IM,IS)	現在の 3 型の時刻を求める.
TIMEC1(CFORM,ITIME)	ITIME を CFORM に従って表現して CFORM で返す.
TIMEC2(CFORM,ITT)	ITT を CFORM に従って表現して CFORM で返す.
TIMEC3(CFORM,IH,IM,IS)	IH,IM,IS を CFORM に従って表現して CFORM で返す.

CFORM 中で使えるキーとなる文字は, 'H' : 時, 'M' : 分, 'S' : 秒, である. たとえば ITIME=100930 に対して CFORM='HH:MM:SS' と指定して TIMEC1 を呼ぶと CFORM='10:09:30' が返される.

8.3 サブルーチンの説明

8.3.1 TIME12/TIME13/TIME21/TIME23/TIME31/TIME32

1. 機能

時刻の型を変換する.
TIME12：1 型の時刻を 2 型の時刻に変換する.
TIME13：1 型の時刻を 3 型の時刻に変換する.
TIME21：2 型の時刻を 1 型の時刻に変換する.
TIME23：2 型の時刻を 3 型の時刻に変換する.
TIME31：3 型の時刻を 1 型の時刻に変換する.
TIME32：3 型の時刻を 2 型の時刻に変換する.

2. 呼び出し方法

```
CALL TIME12(ITIME,ITT)
CALL TIME13(ITIME,IH,IM,IS)
CALL TIME21(ITIME,ITT)
CALL TIME23(IH,IM,IS,ITT)
CALL TIME31(ITIME,IH,IM,IS)
CALL TIME32(IH,IM,IS,ITT)
```

3. パラメーターの説明

ITIME	(I)	1 型の時刻.
IH	(I)	時.
IM	(I)	分.
IS	(I)	秒.
ITT	(I)	通しの秒数.

4. 備考

(a) なし.

8.3.2 TIMEQ1/TIMEQ2/TIMEQ3

1. 機能

現在の時刻を求める.
TIMEQ1：現在の 1 型の時刻を求める.
TIMEQ2：現在の 2 型の時刻を求める.
TIMEQ3：現在の 3 型の時刻を求める.

2. 呼び出し方法

```
TIMEQ1(ITIME)
TIMEQ2(ITT)
TIMEQ3(IH,IM,IS)
```

3. パラメーターの説明

ITIME	(I)	1 型の時刻.
IH	(I)	時.
IM	(I)	分.
IS	(I)	秒.
ITT	(I)	通しの秒数.

4. 備考

   (a) これらはシステムに依存したサブルーチンなので, 正しく移植されていない可能性もある.

8.3.3 TIMEC1/TIMEC2/TIMEC3

1. 機能

   フォーマットを指定して時刻を文字列で表現する.
   TIMEC1 : 1 型の時刻を文字列で表現する.
   TIMEC2 : 2 型の時刻を文字列で表現する.
   TIMEC3 : 3 型の時刻を文字列で表現する.

2. 呼び出し方法

```
CALL TIMEC1(CFORM,ITIME)
CALL TIMEC2(CFORM,ITT)
CALL TIMEC3(CFORM,IH,IM,IS)
```

3. パラメーターの説明

ITIME	(I)	1 型の時刻.
IH	(I)	時.
IM	(I)	分.
IS	(I)	秒.
ITT	(I)	通しの秒数.
CFORM	(C*(*))	時刻のフォーマット. 入力パラメータでもあり, 出力パラメーターでもある. CFORM 中で使えるキーとなる文字は, 'H' : 時, 'M' : 分, 'S' : 秒, である. たとえば ITIME=100930 に対して CFORM='HH:MM:SS' と指定して TIMEC1 を呼ぶと CFORM='10:09:30' が返される.

4. 備考

   (a) 時刻のフォーマットの指定方法について, 正確には, キーとなる文字が最初に現れたところから最後に現れたところの間の部分をおきかえる. したがって例えば, 'HHHHH/MMMM' と書くのと 'HAAAH/MBBM と書くのは同じである.

第 9 章

MISCLIB：雑多な関数 ・サブルーチン *☆

9.1 概要

雑多な関数 ・サブルーチンを集めたパッケージ.

9.2 サブルーチンのリスト

DCLVNM(CV) 電脳ライブラリ」のバージョン名を返す.
CDBLK(CHR) 連続した 2 個以上の空白を 1 個の空白で置き換える.

9.3 関数のリスト

CNS(INS) 調べる値が正なら'N', 負なら'S' を返す.

9.4 サブルーチンの説明

9.4.1 DCLVNM

1. 機能

   電脳ライブラリ」のバージョン名を返す.

2. 呼び出し方法

   ```
   CALL DCLVNM(CV)
   ```

3. パラメーターの説明

   CV (C*(*)) バージョン名を返す文字型の引数.

4. 備考

   (a) バージョン名とは, ふつう, ライブラリを収めたファイルシステムのパス名における末尾部分のことを指す (たとえば /usr/local/src/dcl-5.0 における dcl-5.0). しかし, 実際にどのような文字列が返されるかはシステムに依存する. ただし, 必ず空でない文字列が返される.

9.4.2 CDBLK

1. 機能

   連続した 2 個以上の空白を 1 個の空白で置き換える.

2. 呼び出し方法

   ```
   CALL CDBLK(CHR)
   ```

3. パラメーターの説明

CHR	(C*(*))	操作をほどこす文字列. 入力パラメータでもあり出力パラメータでもある.

4. 備考

   (a) このルーチンでは先行する空白も除去される.

9.5 関数の説明

9.5.1 CSN

1. 機能

   調べる値が正なら'N', 負なら'S' を返す (1 文字の) 文字型関数.

2. 呼び出し方法

   ```
   CNS(INS)
   ```

3. パラメーターの説明

INS	(I)	調べる整数値.
CNS	(C*1)	INS が正なら'N', 負なら'S' を返す長さ 1 の文字型関数.

4. 備考

   (a) INS が 0 のときは空白文字を返す.

第 10 章

CLCKLIB : CPU 時間の取り扱い ☆

10.1 概要

実行時間を扱うサブルーチンパッケージ.

10.2 サブルーチンのリスト

CLCKST	時刻を初期化する.
CLCKGT(X)	CLCKST が呼ばれてからの, CPU 時間を求める.
CLCKDT(DT)	時間の分解能を返す.

10.3 サブルーチンの説明

10.3.1 CLCKST

1. 機能

   時刻を初期化する.

2. 呼び出し方法

   ```
   CALL CLCKST
   ```

3. パラメーターの説明

   なし.

4. 備考

   (a) なし.

10.3.2 CLCKGT

1. 機能

   CLCKST が呼ばれてからの CPU 時間を求める.

2. 呼び出し方法

```
CLCKGT(TIME)
```

3. パラメーターの説明

   TIME (R) CLCKST が呼ばれてからの時間. 単位は秒.

4. 備考

   (a) なし.

10.3.3 CLCKDT

1. 機能

   時間の分解能を返す.

2. 呼び出し方法

```
CALL CLCKDT(DT)
```

3. パラメーターの説明

   DT (R) 時間の分解能. 単位は秒.

4. 備考

   (a) なし.

第 11 章

FIOLIB : ファイルの入出力 ☆

11.1 概要

これは, 固定長の文字列データおよびバイナリデータの読み書きに関するサブルーチンパッケージである. ftp などのファイル転送ソフトを念頭において, 使用する機種によらずファイル操作の一元化を計ろうとするものである. したがってファイルそのものの互換性はないが, 適切な転送手段によってファイルの内容は保存され, 文字は文字レベルで, バイナリはバイナリレベルでハンドリングできることになる.

このパッケージは次のような仕様にもとづいて設計されている.

1. 固定長ファイルの入出力を取り扱う.
2. 入力と出力は同時におこなわない.
3. 出力は順次的におこなう.
4. 入力はレコード番号を指定するランダムアクセスを許す.
5. 行末の改行制御をするかどうかが指定できる.
6. 書き込のためオープンすると既存のファイルは消去される.

このような仕様は, 以下の要請からきている.

- unix 系のマシンでは直接参照入出力を念頭においているので 1 を要請する.
- メインフレーム系のマシンでは順番参照入出力を念頭においている (ファイルシステムの違いから一般に直接参照入出力より順番参照入出力のアクセススピードの方が速い) ので 2, 3 を要請する.
- unix 系のマシンでおこなう直接参照入出力の利点を生かしてランダムアクセスができることを要請する. ただしメインフレーム系の順番参照入出力では出力時にレコード番号を指定するようなやりかたは非現実的であり, 実際われわれの使用形態のほとんどの場合が順番参照的であることから, 4 によって入力時のみ疑似的なランダムアクセスを要請する.
- unix 系の文字入出力を念頭において 5 の要請がある.
- 既存のファイルサイズが上書きするファイルサイズより大きいような直接参照出力をおこなうとき, 以前のファイル内容の一部が残ってしまうので 6 を要請する.

以下のサブルーチン群を使用するにあたっては次の点に注意すること.

- 入出力装置番号の妥当性はチェックしない.
- レコード長の妥当性はチェックしない.
- レコード番号の妥当性はチェックしない.
- 指定できる入出力装置番号は 1 から 99 である.

なお以下の説明で共通してあらわれるコンディションコードは, 正常終了していれば 0 が, そうでなければ 0 以外の整数値が返される.

11.2 サブルーチンのリスト

FCOPEN(IOU,CDSN,NRL,CACT,ICON)	ファイルをオープンする.
FCCLOS(IOU,ICON)	ファイルをクローズする.
FCSLFC(CLX)	行末の改行文字を設定する.
FCLEOL(IOU,LEOL)	行末の改行制御を指定する.
FCNREC(IOU,NREC)	レコード番号を指定する.
FCGETR(IOU,CBUF,ICON)	1 レコードを (文字列で) 読み込む.
FCPUTR(IOU,CBUF,ICON)	1 レコードを (文字列で) 書き出す.
FCGETS(IOU,IBUF,ICON)	1 レコードを (配列で) 読み込む.
FCPUTS(IOU,IBUF,ICON)	1 レコードを (配列で) 書き出す.
FCRWND(IOU,ICON)	リワインドする.

11.3 サブルーチンの説明

11.3.1 FCOPEN

1. 機能

   ファイルをオープンする.

2. 呼び出し方法

```
CALL FCOPEN(IOU,CDSN,NRL,CACT,ICON)
```

3. パラメーターの説明

IOU	(I)	入出力装置番号.
CDSN	(C*(*))	ファイル名.
NRL	(I)	レコード長.
CACT	(C*1)	入出力モードの指定. 読み込みのとき'R', 書き込みのとき'W' を指定する.
ICON	(I)	コンディションコード.

4. 備考

   (a) 書き込のためオープンすると既存のファイルは消去される.

11.3.2 FCCLOS

1. 機能

   ファイルをクローズする.

2. 呼び出し方法

```
CALL FCCLOS(IOU,ICON)
```

3. パラメーターの説明

IOU	(I)	入出力装置番号.
ICON	(I)	コンディションコード.

4. 備考

   (a) なし.

11.3.3 FCSLFC

1. 機能

   行末の改行文字を設定する.

2. 呼び出し方法

```
CALL FCSLFC(CLX)
```

3. パラメーターの説明

CLX	(C*(*))	指定する改行文字.

4. 備考

   (a) 改行文字はふつうシステムごとに定められているが, 特定のシステムを念頭においてファイルを作成するときにこのサブルーチンを使用すればよい.

   (b) FCLEOL の前に呼ばなければならない.

   (c) 改行文字は 2 文字以下でなければならない.

11.3.4 FCLEOL

1. 機能

   行末の改行制御をするかどうかを指定する.

2. 呼び出し方法

```
CALL FCLEOL(IOU,LEOL)
```

3. パラメーターの説明

IOU	(I)	入出力装置番号.
LEOL	(L)	改行制御の指定をする. .TRUE. なら改行制御をする; .FALSE. ならしない.

4. 備考

   (a) FCOPEN の前に呼ばなければならない.

   (b) このルーチンを呼んでいないときの省略値は.FALSE. である. 文字入出力を取り扱うときには, .TRUE. を明示的に指定すべきである.

11.3.5 FCNREC

1. 機能

   レコード番号を指定する.

2. 呼び出し方法

```
CALL FCNREC(IOU,NREC)
```

3. パラメーターの説明

IOU	(I)	入出力装置番号.
NREC	(I)	レコード番号. 1 以上の整数値.

4. 備考

   (a) ここで指定したレコード番号は, 次の FCGETR または FCGETS で有効になる.

   (b) このサブルーチンは読み込みモードでのみ使用できる.

11.3.6 FCGETR

1. 機能

   1 レコードを (文字列で) 読み込む.

2. 呼び出し方法

```
CALL FCGETR(IOU,CBUF,ICON)
```

3. パラメーターの説明

IOU	(I)	入出力装置番号.
CBUF	(C*(*))	読み込むデータレコード.
ICON	(I)	コンディションコード.

4. 備考

   (a) CBUF の長さは NRL (FCOPEN 参照) でなければならない.

   (b) FCNREC によってレコード番号が指定されていなければ, 先頭のレコードから順に読み込む; レコード番号が指定されていれば, 先頭のレコードを 1 として指定されたレコード番号のレコードから読み込む.

11.3.7 FCPUTR

1. 機能

   1 レコードを (文字列で) 書き込む.

2. 呼び出し方法

```
CALL FCPUTR(IOU,CBUF,ICON)
```

3. パラメーターの説明

IOU	(I)	入出力装置番号.
CBUF	(C*(*))	書き込むデータレコード. 長さ NRL (FCOPEN 参照) の文字列で指定する.
ICON	(I)	コンディションコード.

4. 備考

   (a) CBUF の長さは NRL (FCOPEN 参照) でなければならない.

   (b) 書き込みは先頭のレコードから順におこなわれる.

11.3.8 FCGETS

1. 機能

   1 レコードを (配列で) 読み込む.

2. 呼び出し方法

```
CALL FCGETS(IOU,IBUF,ICON)
```

3. パラメーターの説明

IOU	(I)	入出力装置番号.
IBUF	(I)	読み込むデータレコード.
ICON	(I)	コンディションコード.

4. 備考

   (a) IBUF の長さに 4 をかけたものが NRL (FCOPEN 参照) となっていなければならない.

   (b) FCNREC によってレコード番号が指定されていなければ, 先頭のレコードから順に読み込む; レコード番号が指定されていれば, 先頭のレコードを 1 として指定されたレコード番号のレコードから読み込む.

11.3.9 FCPUTS

1. 機能

   1 レコードを (配列で) 書き込む.

2. 呼び出し方法

```
CALL FCPUTS(IOU,IBUF,ICON)
```

3. パラメーターの説明

IOU	(I)	入出力装置番号.
IBUF	(I)	書き込むデータレコード.
ICON	(I)	コンディションコード.

4. 備考

(a) IBUF の長さに 4 をかけたものが NRL (FCOPEN 参照) となっていなければならない.

(b) 書き込みは先頭のレコードから順におこなわれる.

11.3.10 FCRWND

1. 機能

リワインドする.

2. 呼び出し方法

```
CALL FCRWND(IOU,ICON)
```

3. パラメーターの説明

IOU	(I)	入出力装置番号.
ICON	(I)	コンディションコード.

4. 備考

(a) なし.

第 12 章

RANDLIB : 疑似乱数 ☆

12.1 概要

疑似乱数を発生させる関数. いくつかのアルゴリズムによる関数を用意している.

12.2 関数のリスト

RNGU0(ISEED) 一様乱数. システムルーチンを使用する.
RNGU1(ISEED) 一様乱数. 混合合同法.
RNGU2(ISEED) 一様乱数. 混合合同法+シャッフル.
RNGU3(ISEED) 一様乱数. Numerical Recipes の RAN3.

12.3 関数の説明

12.3.1 RNGU0/RNGU1/RNGU2/RNGU3

1. 機能

   [0,1] の一様乱数を発生させる.

2. 呼び出し方法

   ```
   RNGU0(ISEED)
   RNGU1(ISEED)
   RNGU2(ISEED)
   RNGU3(ISEED)
   ```

3. パラメーターの説明

ISEED	(I)	乱数の種. 最初の呼び出しのとき, 0 でない値を与える. ISEED の戻り値は 0 である. 2 回目以降は, この値を変更せずに呼び出す. ルーチンを初期化しなおすときにには再び 0 でない値を与えればよい.

4. 備考

   (a) システム供給ルーチンがない場合には, RNGU0 として RNGU1 等と同等のルーチンが用いられている.

(b) これらのルーチンは W.H.Press 他著 Numerical Recipes (Cambridge University Press) に基づいている. アルゴリズムなどについては, この本を参照のこと.

第 13 章

HEXLIB : 16 進定数の処理

13.1 概要

16 進定数の処理をおこなうサブルーチンパッケージ. 1 語長が 32bit のシステムを念頭においている.

ビットパターンを与える/返す引数は 1 語長であれば, 整数型である必要はない.

13.2 サブルーチンのリスト

HEXDIC(IP,CP)	ビットパターンを 16 進表現の文字列化する.
HEXDCI(CP,IP)	16 進表現の文字列をビットパターン化する.

13.3 サブルーチンの説明

13.3.1 HEXDIC

1. 機能

   ビットパターンを 16 進表現の文字列化する.

2. 呼び出し方法

```
CALL HEXDIC(IP,CP)
```

3. パラメーターの説明

IP	(I)	ビットパターンを調べる 1 語長の引数.
CP	(C*(*))	16 進表現の文字列を返す文字型の引数. '0'- '9', 'A', 'B', 'C', 'D', 'E', 'F' の組合せで返す.

4. 備考

   (a) CP の長さが N (=LEN(CP)) ならば IP の下位 MIN(N*4,NB) ビット (NB は内部変数'NBITSPW' で決まる数) を調べそのパターンを右詰めして返す. N が NB より大きいとき CP(1:N*4-NB) は不定である.

13.3.2 HEXDCI

1. 機能

   16 進表現の文字列をビットパターン化する.

2. 呼び出し方法

   ```
   CALL HEXDCI(CP,IP)
   ```

3. パラメーターの説明

CP	(C*(*))	16 進表現の文字列を与える文字型の引数. '0'- '9', 'A', 'B', 'C', 'D', 'E', 'F' の組合せで表現する.
IP	(I)	ビットパターンを返す 1 語長の引数.

4. 備考

   (a) CP の長さが N (=LEN(CP)) ならば CP の下位 MIN(N,NB/4) 文字 (NB は内部変数'NBITSPW' で決まる数) を調べそのパターンを右詰めして返す. N が NB/4 より大きいとき, 下位 NB/4 文字 (CP(N-NB/4+1:N)) のパターンを調べる.

第 14 章

REALLIB : 実数の変換

14.1 概要

システムに依存する実数表現を解釈する.

それぞれのシステムにおいて表現できる指数範囲について注意すること.

14.2 関数のリスト

R4IBM(RR)	IBM 形式の 4 バイト実数表現を解釈する.
R4IEEE(RR)	IEEE 形式の 4 バイト実数表現を解釈する.

14.3 関数の説明

14.3.1 R4IBM

1. 機能

   IBM 形式の 4 バイト実数表現を解釈して, いま使用しているシステムにおける実数値を返す.

2. 呼び出し方法

   ```
   R4IBM(RR)
   ```

3. パラメーターの説明

RR	(R)	IBM 形式の 4 バイト実数値.
R4IBM	(R)	いま使用しているシステムにおける実数値を返す実数型関数値.

4. 備考

   (a) なし.

14.3.2 R4IEEE

1. 機能

   IEEE 形式の 4 バイト実数表現を解釈して, いま使用しているシステムにおける実数値を返す.

2. 呼び出し方法

   ```
   R4IEEE(RR)
   ```

3. パラメーターの説明

RR	(R)	IEEE 形式の 4 バイト実数値.
R4IEEE	(R)	いま使用しているシステムにおける実数値を返す実数型関数値.

4. 備考

   (a) なし.

謝辞

資源	**dcl-5**
プログラム製作	石岡圭一, 酒井敏, 佐藤薫, 塩谷雅人, 沼口敦
プログラム作成協力	石渡正樹, 竹広真一, 中島健介, 西憲敬, 林祥介, 保坂征宏, 余田成男
デバッグ協力	石田明生, 石渡正樹, 市川香, 沖大幹, 沖理子, 桜井隆博, 佐藤正樹, 竹広真一, 中島健介, 沼口敦, 根田昌典, 野村真佐子, 藤尾伸三, 保坂征宏
文章	酒井敏, 塩谷雅人, 余田成男
文章協力	石渡正樹, 佐藤薫, 佐藤正樹, 竹広真一, 中島健介, 西憲敬, 沼口敦, 林祥介, 保坂征宏
データソース	NCAR [フォント情報, 地図情報]
参考資源	Layout [二木徹], XYGRAPH [佐藤亨], Numerical Recipes [Willam H. Press et al.], NCARG [NCAR], FFTPACK [Paul Swarztrauber]

版権

本資源の版権は地球流体電脳倶楽部に属する. 資源の利用にあたっては地球流体電脳倶楽部の定める規定にしたがっていただきたい. 原則として, 教育的目的の場合には自由に使用・改変して良いものとしている. 地球流体電脳倶楽部が定める規定の詳細は地球流体電脳倶楽部サーバー

```
dennou.gaia.h.kyoto-u.ac.jp
```

上の

```
~ftp/saloon/dennou/To.Users
```

というファイルに収められているので参照されたい. (このファイルは電脳ライブラリを収めたファイルシステムのトップディレクトリにも置かれている.) 上記のアクセスができない場合は, 電子メールにて地球流体電脳倶楽部管理グループ

```
dennou_admin@gaia.h.kyoto-u.ac.jp
```

へ連絡されたい.

引用について

「地球流体電脳ライブラリ」の著作者は「地球流体電脳倶楽部」, その英語名は 'GFD-DENNOU Library', 'GFD-DENNOU Club' である. 地球流体電脳ライブラリを用いて作成された著作物等には, 英語では例えば "The figures were produced by GFD-DENNOU Library." 日本語では例えば「図の作成には地球流体電脳ライブラリをもちいた.」のように引用していただければ幸いである.