PDIC 辞書形式仕様書
第 0.80 版

2003.4.28

SGM00353@niftyserve.or.jp　　TaN

目次

1　はじめに

この仕様書は Personal Dictionary(略して PDIC)という英和辞書アプリケーションで使用している辞書の構造を述べています。PDIC は 2002 年現在、DOS, Windows(3.1, 95, 98, Me, NT4.0, 2000, XP), WindowsCE 上で動作しています。

この辞書の特徴は単一キーで、検索が高速であるところです。登録データはブロックごとに分割した領域に格納することにより、インデックス部を最低限に抑え、コンパクトな辞書となっています。

この仕様書はプログラマ向けのもので、一般ユーザが読むべきものではありません。また、プログラミングに長けている人でもこの仕様書は難解です。というのは、この仕様書をメモ代わりに使用しているため、読者が理解しやすいように、というより、備忘録としているためです。

2　辞書構造概略

2.1　辞書の種類

PDIC には現在公開されているもので４種類存在します。

辞書バージョン	一般名称	対応アプリケーション
1.00	旧辞書形式	DOS 版 PDIC Ver.3.xx～Ver.4.xx
2.00	新辞書形式 1	DOS 版 PDIC Ver.5.xx 以降及び PDIC for Windows(16bit 版,32bit 版)
3.00	新辞書形式 2	PDIC for Windows (32bit 版)
4.00	Hyper 辞書形式	PDIC for Win32 Ver.3.xx 以降
5.00	Hyper 辞書形式	PDIC for Win32 Ver.5.00 以降 （現在 read/write のみ、新規作成はできない）
5.00	Unicode 辞書	PDIC/Unicode

2.2　各辞書形式の内容

辞書形式	辞書サイズ	最大見出語長	最大訳語長	最大用例長	最大発音記号長	他
旧辞書形式 1.00	ｽﾓｰﾙ辞書 ﾐﾃﾞｨｱﾑ辞書 ﾗｰｼﾞ辞書	忘れた	忘れた(^^;	忘れた(^^;		
新辞書形式１ 2.00	ﾀｲﾆｨ辞書 ｽﾓｰﾙ辞書 ﾐﾃﾞｨｱﾑ辞書 ﾗｰｼﾞ辞書	128ﾊﾞｲﾄ	300ﾊﾞｲﾄ	300ﾊﾞｲﾄ		
新辞書形式２ 3.00	ﾀｲﾆｨ辞書 ｽﾓｰﾙ辞書 ﾐﾃﾞｨｱﾑ辞書 ﾗｰｼﾞ辞書	248ﾊﾞｲﾄ	3000ﾊﾞｲﾄ	20,000ﾊﾞｲﾄ (圧縮対応)	1,000ﾊﾞｲﾄ	ﾌｧｲﾙﾘﾝｸｵﾌﾞｼﾞｪｸﾄ OLEｵﾌﾞｼﾞｪｸﾄ 圧縮対応
Hyper辞書形式 4.00	辞書サイズの区別はない	248ﾊﾞｲﾄ	3000ﾊﾞｲﾄ	20,000ﾊﾞｲﾄ (圧縮対応)	1,000ﾊﾞｲﾄ	ﾌｧｲﾙﾘﾝｸｵﾌﾞｼﾞｪｸﾄ OLEｵﾌﾞｼﾞｪｸﾄ 圧縮対応
Hyper辞書形式 5.00	辞書サイズの区別はない	248ﾊﾞｲﾄ	3000ﾊﾞｲﾄ	20,000ﾊﾞｲﾄ (圧縮対応)	1,000ﾊﾞｲﾄ	ﾌｧｲﾙﾘﾝｸｵﾌﾞｼﾞｪｸﾄ OLEｵﾌﾞｼﾞｪｸﾄ 圧縮対応 辞書順辞書対応
Unicode辞書 5.00	辞書サイズの区別はない	248ﾊﾞｲﾄ	3000ﾊﾞｲﾄ	20,000ﾊﾞｲﾄ (圧縮対応)	1,000ﾊﾞｲﾄ	ﾌｧｲﾙﾘﾝｸｵﾌﾞｼﾞｪｸﾄ OLEｵﾌﾞｼﾞｪｸﾄ 圧縮対応 辞書順辞書対応

旧辞書形式は、現在ほとんど用いられていないと思われるので、この辞書形式についての説明は省略します。

この仕様書では、新辞書形式１、２及びHyper辞書形式について説明します。

ここからは、旧辞書形式を **OLDDIC**, 新辞書形式１を **NEWDIC**, 新辞書形式２を **NEWDIC2**, Hyper辞書形式を **NEWDIC3** と略します。

NEWDIC2はNEWDICの単純な拡張であるため、NEWDICとNEWDIC2は同時に説明し、異なる部分のみ補足していますが、NEWDIC3は、NEWDIC2と若干構造が異なるため、別の章を設けて説明してあります。Unicode辞書も別の章にて説明してあります。

2.3　本仕様書を読むための注意事項

以降現れる C 言語の型宣言などは次のようになっています。

```
typedef unsigned char uchar;
typedef unsigned char byte;
typedef unsigned short ushort;
typedef unsigned short word;
typedef unsigned int uint;
typedef unsigned long ulong;
```

3　NEWDIC,NEWDIC2 の辞書構造

3.1　辞書の全体の構成

ヘッダー部
固定長(256ﾊﾞｲﾄ)
インデックス部
固定長（辞書サイズによって異なる） ※プログラム内部では、このインデックス部をさらに 2048ﾊﾞｲﾄのブロック単位で管理している。
データ部
固定ブロック長単位でブロック数は可変 ブロック長は辞書サイズによって異なる 新辞書形式２ではさらにブロックの連結を行って実質的な可変長ブロックとなっているので注意

※Hyper 辞書形式はこれとは若干異なります。次章参照

3.2 ヘッダー部

ヘッダー部は256ﾊﾞｲﾄの固定長
以下はソースファイルからの引用で､新辞書形式２はNEWDIC2のdefineを使用しています。
注意：ここで述べているデータ部のブロック番号はすべて物理ブロック番号を示しています。
（物理ブロック番号についてはインデックス部を参照）

```
#define   L_HEADERNAME  100       // ヘッダー部文字列長
#define   L_DICTITLE         40         // 辞書タイトル名長

struct HEADER {
     char headername[ L_HEADERNAME ]; // 辞書ヘッダータイトル
     char dictitle[ L_DICTITLE ];      // 辞書名
     short version;               // 辞書のバージョン
     short lword;                 // 見出語の最大長
     short ljapa;                 // 訳語の最大長
     short block_size;            // １ブロックのバイト数
                                       // NEWDIC2の場合は、/16
     short index_block;           // インデックスブロック数
     short header_size;           // ヘッダーのバイト数
     unsigned short index_size;   // インデックスのバイト数

     short empty_block;           // 空きブロックの先頭物理ブロック番号（ない
ときは-1）
     short nindex;                // インデックスの要素の数
     short nblock;                // 使用データブロック数
     unsigned long nword;         // 登録単語数

     byte dicorder;               // 辞書の順番
     byte dictype;                // 辞書の種別

     byte attrlen;                // 単語属性の長さ
#ifdef NEWDIC2
     long olenumber;                   // OLE用シリアル番号
     byte os;                     // OS
     // 以下の言語IDは未決定
     ushort lid_word;             // 見出語言語ID
     ushort lid_japa;             // 訳語部言語ID
     ushort lid_exp;                   // 用例部言語ID
     ushort lid_pron;             // 発音記号言語ID
     ushort lid_other;            // その他言語ID
//   char dummy[ 84 ];            // 言語IDを使用しないときのdummy
     char dummy[ 74 ];            // 言語IDを使用したときのdummy
#else
     char dummy[ 89 ];
#endif
};
```

<table>
<tr><th>名前</th><th>型</th><th>説明</th></tr>
<tr><td>headername</td><td>char
*100</td><td>ヘッダー文字列。とくに意味はないが、以下のような文字列が必ず入る。
NEWDIC:
¥x1b[2J¥x1b[32;7m ================= ＰＤＩＣ用 辞書 =================
¥x1b[37;0m¥x1a
NEWDIC2:
¥x1b[2J¥x1b[32;7m ===============
Dictionary for PDIC ===============
¥x1b[37;0m¥x1a</td></tr>
<tr><td>dictitle</td><td>char
*40</td><td>辞書名。現在は未使用で NULL でパディング</td></tr>
<tr><td>version</td><td>short</td><td>辞書のバージョン。上位ﾊﾞｲﾄはメジャー番号、下位ﾊﾞｲﾄはマイナー番号
NEWDIC:0x200
NEWDIC2:0x300
バージョン番号の取り扱いの原則は、0x100 以上異なる場合は、取り扱い不可能、0x010 以上異なる場合は、読み取りのみ可能、0x001 の場合は読み取り、書き込み可能であると判断。例えば、新規作成すると 0x300 の辞書を生成する PDIC の場合、0x301 の辞書は読み書き可能であると判断でき、0x0310 は読み取りのみ可能な動作が行える。しかし、現在公開されている PDIC ではこの条件にしたがったものは今のところない。</td></tr>
<tr><td>lword</td><td>short</td><td>見出語の最大長。現在未参照。
NEWDIC:128
NEWDIC2:248</td></tr>
<tr><td>ljapa</td><td>short</td><td>訳語の最大長。現在未参照。
NEWDIC:300
NEWDIC2:3000</td></tr>
<tr><td>block_size</td><td>short</td><td>データブロックの１ブロックの長さを示す。
NEWDIC2 ではこの値をさらに 16 で割った値。
<table><tr><th></th><th>NEWDIC</th><th>NEWDIC2</th></tr><tr><td>ﾀｲﾆｨ辞書
ｽﾓｰﾙ辞書
ﾐﾃﾞｨｱﾑ辞書
ﾗｰｼﾞ辞書</td><td>1024
2048
4096
8192</td><td>64
128
256
512</td></tr></table></td></tr>
<tr><td>index_block</td><td>short</td><td>インデックスのサイズをブロック数であらわしたもの。
１ブロックは 2048 バイト
<table><tr><td>ﾀｲﾆｨ辞書
ｽﾓｰﾙ辞書
ﾐﾃﾞｨｱﾑ辞書
ﾗｰｼﾞ辞書</td><td>1
5
10
20</td></tr></table></td></tr>
<tr><td>header_size</td><td>short</td><td>ヘッダーのサイズ。256 固定</td></tr>
</table>

index_size	short	インデックス部のサイズ。	
		ﾀｲﾆｨ辞書 ｽﾓｰﾙ辞書 ﾐﾃﾞｨｱﾑ辞書 ﾗｰｼﾞ辞書	2048 10240 20480 40960
empty_block	short	先頭の空きデータブロック番号(0～4095) 存在しないときは-1	
nindex	short	インデックスの要素数(最大 4096)(要素については後述)	
nblock	short	使用データブロック数	
nword	ulong	登録単語数	
dicorder	byte	見出語の登録順(NEWDIC2 のみ) 現在の PDIC はコード順のみをサポート	
		0	コード順
		1	大文字・小文字同一視順
		2	辞書順
		3	降順
dictype	byte	辞書の属性(NEWDIC2 のみ) 以下のフラグの組み合わせ	
		0x01	LHA 方式の圧縮を利用する
		0x10	Unicode を使用している(未完成)
		0x20	多国語辞書(未完成)
attrlen	byte	属性長(１に固定)	
olenumber	long	最新の OLE ｵﾌﾞｼﾞｪｸﾄ番号(NEWDIC2 のみ)	
os	byte	使用 OS(NEWDIC2 のみ) 以下のいずれかの値	
		0x00	DOS,Windows,OS/2(x86 系)
		0x01	Mac(68k 系)
		0x02	UNIX(?)
lid_word	ushort	見出言語 ID(NEWDIC2 のみ)現在未使用	
lid_japa	ushort	訳語言語 ID(NEWDIC2 のみ)現在未使用	
lid_exp	ushort	用例言語 ID(NEWDIC2 のみ)現在未使用	
lid_pron	ushort	発音記号言語 ID(NEWDIC2 のみ)現在未使用	
lid_other	ushort	その他言語 ID(NEWDIC2 のみ)現在未使用	
dummy	byte	ヘッダー部パディング	

3.3　インデックス部

3.3.1 インデックス部のサイズ

インデックス部ではデータ部の物理的な位置を管理しています。
インデックス部の長さは固定長ですが、辞書サイズによって異なります。

辞書サイズ	ｲﾝﾃﾞｯｸｽｻｲｽﾞ(byte 単位)
ﾀｲﾆｨ辞書	1024
ｽﾓｰﾙ辞書	2048
ﾐﾃﾞｨｱﾑ辞書	4096
ﾗｰｼﾞ辞書	8192

3.3.2 インデックス部の構成

物理ﾌﾞﾛｯｸ番号 2ﾊﾞｲﾄで 0 から始まる	ﾌﾞﾛｯｸ先頭の見出語 可変長で NULL 終端	----------	インデックス部の最後 NULL３ﾊﾞｲﾄ以上であらわす

物理ﾌﾞﾛｯｸ番号とは、ﾃﾞｰﾀ部のﾌﾞﾛｯｸ番号であり、実際のディスク上の位置を示します。
物理ﾌﾞﾛｯｸ番号と見出語の組み合わせをインデックス要素と呼びます。このインデックス要素は、見出語のコード順に並びます。(ただし、ヘッダー部の dicorder がコード順でない場合は、この限りではない) 見出語は、データ部の各ブロックの先頭の見出語をそのまま記憶します。
　インデックス要素は先頭から論理ﾌﾞﾛｯｸ番号という連続した番号をふります。(０～)ただし、これはプログラム処理上に便宜上用いる値ですので、実際の辞書には存在しません。
注意：物理ﾌﾞﾛｯｸ番号の最大値は 4095 に制限されています。したがって、論理ﾌﾞﾛｯｸ番号も 4095 に制限されます。これ以上登録すると NEWDIC2 では正常に動作しなくなります。これは物理ﾌﾞﾛｯｸ番号にﾌﾞﾛｯｸ数の情報も含まれているためです。（後述）

3.3.3 物理ブロック番号からディスク上の位置を求める

物理ﾌﾞﾛｯｸ番号からディスク上の位置へ変換するには以下の式を用います。

(ﾃﾞｨｽｸ上の位置)=(ﾍｯﾀﾞｰｻｲｽﾞ)+(ｲﾝﾃﾞｯｸｽｻｲｽﾞ)+(ﾃﾞｰﾀﾌﾞﾛｯｸ長)*(物理ﾌﾞﾛｯｸ番号)

ｲﾝﾃﾞｯｸｽｻｲｽﾞ、ﾃﾞｰﾀﾌﾞﾛｯｸ長は辞書サイズによって異なることに注意してください。
さらに NEWDIC2 では、インデックス部の物理ﾌﾞﾛｯｸ番号の上位４ビットで(ﾌﾞﾛｯｸ数)-1 を表す。例えば、物理ﾌﾞﾛｯｸ番号が 0x2011 となっていた場合、ﾃﾞｰﾀﾌﾞﾛｯｸが 17 から始まり、３ﾌﾞﾛｯｸ使用していることを示します。この３つの使用ﾌﾞﾛｯｸは物理的に連続していなければなり

ません。

作者談：NEWDIC2 ではだいぶ複雑な構成となってしまいました。OLE ｵﾌﾞｼﾞｪｸﾄなど大きなデータを記録するには最低でも 64KB 必要と考えたのですが、NEWDIC の仕様では 8KB が限界で拡張する必要がありました。単純にﾃﾞｰﾀﾌﾞﾛｯｸサイズを 64KB にすれば解決したのでしょうが、そうすると検索が遅くなる、辞書の格納率が悪くなどの理由から、現在のような擬似可変長ﾌﾞﾛｯｸとなった、と記憶してます。(実際はもっと深刻な問題があったような…)

3.4　データブロック部

データ部には見出語、訳語、用例などの実際のデータが記憶されています。
データ部の構成には大きく分けて２種類有ります。見出語と訳語のみでよい場合と、それ以外の用例、発音記号などを必要とする場合があります。便宜上、前者を基本構成、後者を拡張構成と呼びますが、拡張構成は基本構成に拡張部を付け加えた形になっています。基本構成と拡張構成の区別はこれから述べる見出語属性のフラグによって判断します。

3.4.1 基本構成

基本構成

名称	サイズ	備考
ﾌｨｰﾙﾄﾞ長	2ﾊﾞｲﾄ	見出語から訳語の最後までのﾊﾞｲﾄ単位の長さ(ﾌｨｰﾙﾄﾞ長、圧縮長は含めない)
圧縮長	1ﾊﾞｲﾄ	見出語部の圧縮している長さを示す
見出語	可変長 NULL 終端	圧縮してある
見出語属性	1ﾊﾞｲﾄ	後述
訳語	可変長 NULL 終端**無し**	

見出語部は圧縮されています。圧縮は同一のﾃﾞｰﾀﾌﾞﾛｯｸにおいて、直前の見出語との差分を取ることによって行っています。例えば、ﾃﾞｰﾀﾌﾞﾛｯｸの先頭の見出語が ABC であり、次の見出語が ABDEF だった場合は、２番目の見出語部に入る文字列は ABDEF ではなく、DEF となり、圧縮長は２となります。さらに ABDG という見出語が続いた場合は、圧縮長が３となり、見出語部に入る文字列は G となります。
ﾃﾞｰﾀﾌﾞﾛｯｸ先頭の見出語の圧縮長は必ず零になります。

見出語圧縮の例

見出語	圧縮長	圧縮後の見出語
ABC	0	ABC
ABDEF	2	DEF
ABDG	3	G

見出語属性

次のフラグの組み合わせです。

数値	意味

0x80	必ずつける
下位 3bit	単語レベル値(0-7)
0x10	拡張構成フラグ
0x20	暗記必須単語フラグ
0x40	修正単語フラグ

0x10 の拡張構成がある場合は、次節の拡張構成になります。

3.4.2 拡張構成

名称	サイズ	備考
ﾌｨｰﾙﾄﾞ長	2ﾊﾞｲﾄ	見出語から拡張終了までのﾊﾞｲﾄ単位の長さ(ﾌｨｰﾙﾄﾞ長､圧縮長は含めない)
圧縮長	1ﾊﾞｲﾄ	見出語部の圧縮している長さを示す
見出語	可変長 NULL 終端	
見出語属性	1ﾊﾞｲﾄ	前述
訳語	可変長 NULL 終端**あり**	
拡張属性 1	1ﾊﾞｲﾄ	後述
拡張内容 1	可変長	拡張属性によって内容は異なる
拡張属性 2	1ﾊﾞｲﾄ	
拡張内容 2	可変長	
拡張終了	1ﾊﾞｲﾄ	0x80

拡張部の特徴としては、バイナリデータ、データの圧縮などが可能になっています。また、拡張属性にはまだ未使用のフラグが残っているため､今後更に拡張される可能性があります。

拡張属性

数値	
0x01	用例
0x02	発音記号
0x03	未定義
0x04	ﾘﾝｸﾃﾞｰﾀ

0x05～0x0f	未定義
0x10	バイナリデータフラグ
0x20	未定義
0x40	圧縮フラグ
0x80	拡張終了

拡張属性は下位 4bit が項目の種類、上位 4bit が何らかのフラグであることを示します。
フラグによってさらに、拡張部の構成は異なります。
拡張部の構成は非常に複雑です。もし、すべての拡張部を必要としない場合は、0x10 のフラグだけを見て、スキップすることができます。つまり、0x10 がなければ NULL 終端テキストであると判断し、0x10 がある場合は指定されたサイズだけスキップすれば済みます。

拡張部の構成１-バイナリデータフラグ、圧縮フラグ、拡張終了フラグがない場合

この場合、NULL 終端のあるテキストデータであると判断します。

拡張属性	1ﾊﾞｲﾄ	0x00～0x0f
テキストデータ	NULL 終端有り	

圧縮されていない用例や、発音記号はこのフォーマットです。

拡張部の構成２-バイナリデータフラグのみがある場合

拡張属性	1ﾊﾞｲﾄ	0x10～0x1f
サイズ	2ﾊﾞｲﾄ	拡張データの長さをﾊﾞｲﾄ単位で示す
拡張データ	可変長	

圧縮データや、ﾘﾝｸﾃﾞｰﾀがこのフォーマットになります。

拡張部の構成３-圧縮フラグありの場合

拡張属性	1ﾊﾞｲﾄ	0x50～0x5f
サイズ	2ﾊﾞｲﾄ	非圧縮長から圧縮データの最後までのﾊﾞｲﾄ単位の長さを示す
非圧縮長	1ﾊﾞｲﾄ	非圧縮データのﾊﾞｲﾄ単位の長さ
非圧縮データ	可変長	
圧縮ヘッダー	可変長	
圧縮データ	可変長	

圧縮ありでは、必ずバイナリデータフラグが必要です。したがって、0x50～という拡張属性になります。

非圧縮長は1ﾊﾞｲﾄであらわすため、255ﾊﾞｲﾄを超える長さにはできないことに注意してください。
圧縮内容のフォーマットは省略 - 次回に譲る

3.4.3 ﾘﾝｸﾃﾞｰﾀの構成

拡張属性にﾘﾝｸﾃﾞｰﾀがある場合はさらにフォーマットが定められています。

ﾘﾝｸﾃﾞｰﾀの構成 1-非圧縮時

拡張属性	1ﾊﾞｲﾄ	0x14
サイズ	2ﾊﾞｲﾄ	
リンクタイプ	1ﾊﾞｲﾄ	
ID	4ﾊﾞｲﾄ	
タイトル	可変長 NULL 終端有り	
データ部	可変長	

リンクタイプ

数値	意味	
0x00	未定義	
0x01	OLE	
0x02	ファイルリンク	
0x03	音声データ	
0x04～0xff	未定義	

ﾘﾝｸﾃﾞｰﾀの構成 2-圧縮時

拡張属性	1ﾊﾞｲﾄ	0x14
サイズ	2ﾊﾞｲﾄ	
非圧縮長	1ﾊﾞｲﾄ	リンクタイプからタイトルの最後までのﾊﾞｲﾄ単位の長さ
リンクタイプ	1ﾊﾞｲﾄ	
ID	4ﾊﾞｲﾄ	
タイトル	可変長 NULL 終端有り	
圧縮ヘッダー	可変長	
データ部(圧縮データ)	可変長	

さらにさらに、ﾘﾝｸﾃﾞｰﾀのデータ部はリンクタイプによって異なります。
ここではデータ部のみを示します。
ﾘﾝｸﾃﾞｰﾀの構成 1 -OLE データの場合

省略

ﾘﾝｸﾃﾞｰﾀの構成２-ファイルリンクデータの場合

省略

例：見出語、訳語、用例がある場合

名称	サイズ	備考
ﾌｨｰﾙﾄﾞ長	2ﾊﾞｲﾄ	見出語から訳語の最後までのﾊﾞｲﾄ単位の長さ
見出語	可変長 NULL 終端	
見出語属性	1ﾊﾞｲﾄ	
訳語	可変長 NULL 終端無	
拡張属性(用例)	1ﾊﾞｲﾄ	
用例	可変長 NULL 終端	
拡張終了		

4　NEWDIC3(Hyper 辞書形式) Ver.4.00 の構造

Hyper 辞書形式は、従来の辞書(NEWDIC2)の欠点を改善したものです。

・tiny,small,medium,large という複数の辞書サイズがあり、面倒だった

→１種類で統一

・登録できる１つの単語のデータが最大 64Kﾊﾞｲﾄまでという制限があった

→8Mﾊﾞｲﾄまで可能(この拡張により、フィールド部の構成が変更されています)

・ラージ辞書の場合、データブロックが 8,192ﾊﾞｲﾄ単位であるため、効率が悪かった

→256ﾊﾞｲﾄ単位に

・ラージ辞書でも登録単語数が最大 45 万語程度であった

→実質的な制限はなくなった(20 億語まで 2^31 語)

・新規作成時の辞書サイズが大きかった

→新規作成時、256ﾊﾞｲﾄであり、インデックス部も可変長となったため、無駄が少ない

・ヘッダー部を拡張可能に

・インデックス部を可変長に

・格納率が常に最大になるようにアルゴリズムを最適化(PDIC/W32)

・ファイルサイズは最大 500G バイトまで(ただし、PDIC では 2G まで)

・単語レベルを 16 段階に

注意：現在この辞書が主流ですが、Ver.4.00 はヘッダー部のアライメントが取れていないため、徐々に Ver.5.00 へ移行する予定です。互換性を考えないような状況では Ver.5.00 で作成するようにしてください。詳細は次の章で説明します。

4.1　辞書の全体の構成

ヘッダー部
固定長(256ﾊﾞｲﾄ)
拡張ヘッダー部
可変長(最大 4G バイト) 現在のところここは使用していないので通常は０ﾊﾞｲﾄ
インデックス部
可変長(256ﾊﾞｲﾄ単位) 何もデータが無いと０ﾊﾞｲﾄ。最大 256×32,767 = 8M
データ部
可変長(256ﾊﾞｲﾄ単位) 256ﾊﾞｲﾄを１物理ブロック単位とする インデックス部により論理ブロック番号から物理ブロック番号にマッピング

される。１つの論理ブロックには連続した複数の物理ブロックを示すことができる。この場合の連続ブロック数は 32,767 個までなので、32,767×256=8,388,352 バイトが最大の１つの論理的なブロックとなる。物理ブロック番号は最大 2,147,418,112(2G)。

4.1.1 ツリービューモード

Hyper 辞書形式にはツリービューモード用辞書形式というものがあります。中身はほとんど通常の英和辞書と同じなのですが､見出語にツリー構造を記録するという方式を用いているため、通常の辞書と多少異なる処理が必要となります(見出語部の文字列置換など)。そのためにヘッダー部の dictype フィールドに区別するためのフラグを設けています。

もう一点、ツリービューモードでは単にツリー構造を変更する場合は、見出語だけを rename すればいいのですが、PDIC のデータ部の構造上、訳語も全て更新する必要があります。Field2 のような 64KB を超えるデータではこのオーバヘッドが問題となってしまいます。そのために、ツリービューモード用の辞書では、Field2 のデータブロック部の構造も変更を加え、データ部に見出語情報を記録しないようにしています。これにより、単純にインデックス部だけを更新すればよいことになります。(Field2 データは１データブロックにつき１つの見出語しか登録しない、という原則があるので)

Hyper 辞書形式公開前にこの問題が分かっていれば、どのモードでもそうしたのですが・・・。

※ツリービューモードは構想段階で、現在諦めかけています。

4.2　ヘッダー部

ヘッダー部は256ﾊﾞｲﾄの固定長
注意：ここで述べているデータ部のブロック番号はすべて物理ブロック番号を示しています。
（物理ブロック番号についてはインデックス部を参照）

```
#define   L_HEADERNAME  100       // ヘッダー部文字列長
#define   L_DICTITLE         40        // 辞書タイトル名長

struct HEADER {
     char headername[ L_HEADERNAME ]; // 辞書ヘッダータイトル
     char dictitle[ L_DICTITLE ];      // 辞書名
     short version;               // 辞書のバージョン
     short lword;                 // 見出語の最大長
     short ljapa;                 // 訳語の最大長
     short block_size;            // １ブロックのバイト数(256固定)
     short index_block;           // インデックスブロック数
     short header_size;           // ヘッダーのバイト数
     unsigned short index_size;   // インデックスのバイト数(未使用)

     short empty_block;           // 空きブロックの先頭物理ブロック番号（ない
ときは-1）
     short nindex;                // インデックスの要素の数(未使用)
     short nblock;                // 使用データブロック数(未使用)
     unsigned long nword;         // 登録単語数

     byte dicorder;          // 辞書の順番
     byte dictype;                // 辞書の種別

     byte attrlen;                // 単語属性の長さ
     long olenumber;              // OLE用シリアル番号
     byte os;                     // OS
     ushort lid_word;             // 見出語言語ID
     ushort lid_japa;             // 訳語部言語ID
     ushort lid_exp;              // 用例部言語ID
     ushort lid_pron;             // 発音記号言語ID
     ushort lid_other;            // その他言語ID
     ulong extheader;             // 拡張ヘッダーサイズ
     long empty_block2;           // 空きブロック先頭物理ブロック番号
     long nindex2;                // インデックス要素の数
     long nblock2;                // 使用データブロック数
     byte index_blkbit;           // 0:16bit, 1:32bit
     byte reserved[8+1];
     ulong update_count;          // 辞書更新回数
     byte charcode;          // 文字コード
     char dummy[ 43 ];            // 宇宙のバランスを取るためのパディング
};
```

名前	型	説明
headername	char *100	ヘッダー文字列。とくに意味はないが、以下のような文字列が必ず入る。 PDIC 自体はまったく見ていないので、Copyright の表示に使用するもよし、メッセージを書き込むもよし。 ¥x1b[2J¥x1b[32;7m ========== Dictionary for PDIC ========== ¥x1b[37;0m¥x1a
dictitle	char *40	辞書名。現在は未使用で NULL でパディング いずれ PDIC 側からどんな辞書であるか一目で分かるようにするつもりではいるが・・・。
version	short	辞書のバージョン。上位ﾊﾞｲﾄはメジャー番号、下位ﾊﾞｲﾄはマイナー番号 0x0400 バージョン番号の取り扱いの原則は、0x100 以上異なる場合は、取り扱い不可能、0x010 以上異なる場合は、読み取りのみ可能、0x001 の場合は読み取り、書き込み可能であると判断。例えば、新規作成すると 0x300 の辞書を生成する PDIC の場合、0x301 の辞書は読み書き可能であると判断でき、0x0310 は読み取りのみ可能な動作が行える。
lword	short	見出語の最大長。現在未参照。 248
ljapa	short	訳語の最大長。現在未参照。 3000
block_size	short	データブロックの１ブロックの長さを示す。 256 に固定 ※ここの値を変更すれば１見出語あたりの登録データを大きくすることが可能です。(PDIC はそうなるように作っていますが、動作未確認)
index_block	short	インデックスのサイズをブロック数であらわしたもの。 １ブロックは 256 バイト インデックス部のサイズは、index_block * block_size で求めること。 このフィールドは NEWDIC,NEWDIC2 と名前が同じでも扱いがまったく異なるので注意が必要。
header_size	short	ヘッダーのサイズ。256 固定
index_size	short	インデックス部のサイズ。 NEWDIC3 では参照してはいけない！！ index_block * block_size で求めること！！
empty_block	short	先頭の空きデータブロック番号 NEWDIC3 では empty_block2 を使用すること！！
nindex	short	インデックスの要素数(要素については後述) NEWDIC3 では nindex2 を使用すること！！
nblock	short	使用データブロック数

		NEWDIC3 では nblock2 を使用すること！！	
nword	ulong	登録単語数	
Dicorder	byte	見出語の登録順 現在の PDIC はコード順のみをサポート	
		0	コード順
		1	大文字・小文字同一視順
		2	辞書順
		3	降順
Dictype	byte	辞書の属性 以下のフラグの組み合わせ	
		0x01	AR 方式の圧縮を利用する
		0x10	Unicode を使用している
		0x20	（多国語辞書）
		0x40	この辞書は暗号化されている
		0x80	ツリービューモード用の辞書
Attrlen	byte	属性長(１に固定)	
Olenumber	long	最新の OLE ｵﾌﾞｼﾞｪｸﾄ番号	
Os	byte	使用 OS（この辞書が使用している文字コード体系）	
		0x00	DOS,Windows,OS/2(x86 系) - SJIS，改行は CR,LF
lid_word	ushort	見出語言語 ID 現在未使用	
lid_japa	ushort	訳語言語 ID 現在未使用	
lid_exp	ushort	用例言語 ID 現在未使用	
lid_pron	ushort	発音記号言語 ID 現在未使用	
lid_other	ushort	その他言語 ID 現在未使用	
extheader	ulong	拡張ヘッダーサイズ(バイト単位)	
empty_block2	long	先頭空きブロック番号 ない場合は-1(0xFFFFFFFF)	
nindex2	ulong	インデックス要素の数	
nblock2	ulong	使用データブロック数	
index_blkbit	byte	インデックス部の物理ブロック番号のビット数(0:16bit,1:32bit)	
cypt	8byte	暗号コード	
update_count	ulong	辞書更新回数	
dummy	49byte	ヘッダー部パディング	

※PDIC 用の辞書であるかどうかの判断は、“固定”と記してある値を判断するのが適当だと思います。特別な ID というものはありませんので。

4.3 暗号化のセキュリティについて

暗号化された場合、インデックス部以外が暗号化されます。このセキュリティレベルは、プ

ログラミングの知識が無い人にはまったく不可能、一般のプログラマーでも不可能、一流のクラッカーであれば解読可能なレベルです（完全ではありません）。
具体的にはインデックス部をバイナリエディターで見ることができるので、どんな辞書であるかはだいたい予想はできますが、その中身については暗号化アルゴリズムがわからない限りどんなクラッカーであっても解読は不可能です。（コンピューターが高速化されても恐らく無理）アルゴリズムを知ったとしてもパスワードの工夫によってさらに解読を難しくすることも可能です。
以上から、国家機密、企業の重要情報を維持するだけのセキュリティレベルはありません。高レベルのセキュリティを必要とする場合は、辞書の暗号化以外のセキュリティ対策が必要です。
この暗号化によって得られるセキュリティは、辞書を一般公開し、その辞書の価値が一流のクラッカーによって解読に一週間以上費やすだけの価値を持つ場合に有効です。
なお、この暗号化は安全である保証をするものではありません。あくまで自分の能力に基づいた予測に過ぎません。

4.4　拡張ヘッダー部

ヘッダー部の extheader がゼロ以外の場合、拡張ヘッダーが存在します。拡張ヘッダー部はヘッダー部の直後に配置されます。

extheader ではバイト単位でサイズを指定できますが、256 バイト単位でなければなりません。256 バイト単位に満たない部分はゼロでパディングされます。（PDIC は 256 バイト単位でなくても動作するはずですが、効率が悪くなります）

4.4.1 拡張ヘッダー部の構成

サイズ（２バイト）	タグ名（可変長、NULL 終端）	データ（可変長）

サイズはバイト単位で、タグ名とデータの合計サイズ。

タグ名は英数字 _ - のみ使用可能。

サイズがゼロの場合は拡張ヘッダーの終端。extheader はこの終端までのサイズを示す。

タグ名は最大 40 文字まで。

タグ名の重複が可能。

大文字・小文字を区別する。

タグ名は自由に使えますが、pdic- で始まるタグ名は PDIC で予約されています。

拡張ヘッダー部は暗号化の対象になりません。必要な場合は個々のタグで対処する必要があります。

タグ名は辞書の文字コードの種類と一致させる必要がありますが、データ部の文字列は自由です。

4.5　インデックス部

4.5.1 インデックス部のサイズ

インデックス部ではデータ部の物理的な位置を管理しています。
インデックス部は 256 ﾊﾞｲﾄ単位で可変長です。データがまったく登録されていない場合は、0ﾊﾞｲﾄで、最大 256×32,767＝8M バイトまでです。インデックス部のサイズを制限するものはヘッダー部にあるインデックスブロック数(index_block)です。

4.5.2 インデックス部の構成

物理ﾌﾞﾛｯｸ番号	ﾌﾞﾛｯｸ先頭の見出語		インデックス部の最後
2 バイト or 4 バイト	可変長		4 バイト
0 から始まる物理ブロック番号を示す	NULL 終端		NULL 4ﾊﾞｲﾄ以上であらわす

物理ブロック番号とブロック先頭の見出語の対がいくつか並び、最後にインデックス部の最後を示す 4 バイト以上の NULL があります。インデックス部は 256 バイト単位の可変長であるため、残りは NULL でパディングされています。
物理ﾌﾞﾛｯｸ番号とは、ﾃﾞｰﾀ部のﾌﾞﾛｯｸ番号であり、実際のディスク上の位置を示します。
物理ﾌﾞﾛｯｸ番号と見出語の組み合わせをインデックス要素と呼びます。このインデックス要素は、見出語のコード順に並びます。(ただし、ヘッダー部の dicorder がコード順でない場合は、この限りではない)見出語は、データ部の各ブロックの先頭の見出語をそのまま記憶します。
　インデックス要素は先頭から論理ﾌﾞﾛｯｸ番号という連続した番号をふります。これは便宜上のものですが、プログラムする上では必要な概念です。
　インデックス部の物理ブロック番号は、2 バイトの場合と、4 バイトの場合が存在します。どちらを使用するかはインプリメンテーション依存です。PDIC では、すべての物理ブロック番号が 16bit で表せられる場合は 2 バイトを、それ以上の場合は 4 バイトを使用しています。物理ブロック番号が 16bit を超える場合というのは、256×65535=16M バイト以上ということですので、ほとんどの場合は 2 バイトで十分であるといえます。このフィールドが 2 バイトであるか 4 バイトであるかは、ヘッダー部の index_blkbit で示します。

4.5.3 物理ブロック番号からディスク上の位置を求める

物理ﾌﾞﾛｯｸ番号からディスク上の位置へ変換するには以下の式を用います。

(ﾃﾞｨｽｸ上の位置)=(ﾍｯﾀﾞｰｻｲｽﾞ)++(拡張ﾍｯﾀﾞｰｻｲｽﾞ)+(ｲﾝﾃﾞｯｸｽﾌﾞﾛｯｸ数)×(ブロックサイ

ズ)+(ブロックサイズ)*(物理ﾌﾞﾛｯｸ番号)
つまり、
(ディスクの位置)=(header_size) +(extheader)+ (index_block)×(block_size) + (block_size)×(物理ブロック番号)
ブロックサイズは 256 バイトです。

インデックス部で注意するところは、NEWDIC3 では可変長である点です。インデックス部のすぐ後ろがデータ部ですから、インデックス部を拡張したい場合、データ部の先頭のブロックを移動する必要があることです。さらに、データ部のオフセットがずれたり、物理ブロック番号もずれます。そして最悪なことに空きブロックのリンクは絶対物理ブロック番号でリンクしているため、すべて更新する必要があります。PDIC の場合は、空きブロックを必要とするとき、すべての空きブロックリンクを読み込み、メモリ内部で相対物理ブロック番号による片方向リンクリストで管理するため、あまり気にすることなく実装できています。通常、インデックスの更新処理が必要なときは、空きブロックリンクをたどることが多い、ということから重要な問題ではないと思います(今後ともそうであることを願いたい(^^;)。また、プログラム内部で絶対物理ブロック番号を保持している場合も、それらをすべて変更しなければならないということも忘れないでください。

※この仕様は、PDIC のプログラムを簡略化するためです。本当は相対物理ブロック番号のほうが smart であるのは間違いないのですが…。

※また、空きブロックリンクはたとえ相対物理ブロック番号でやっていたとしても、一括登録や一括削除では問題が生じます。何が原因かというと、空きブロックが多数出ることが予想されることです。そうなると、空きブロック更新だけで、かなり時間がかかります。PDIC ではそんなことはないって？実は PDIC では、あるタイミングで空きブロックを無くす処理をやっています。この処理は空きブロックをなくすと同時に辞書サイズも小さくしてくれます。この処理は「重そう」と思うかもしれませんが、意外と軽く、まとめてやるより、少しずつ暇があったら辞書を“簡単に”最適化したほうが速度、サイズともに有利であることがわかりました。

4.6　データブロック部

データ部には見出語、訳語、用例などの実際のデータが記憶されています。データ部は 256 バイト単位で管理され、これをブロックと呼びます。

データ部の種類には「使用ブロック」と「空きブロック」があります。空きブロックはヘッダー部の empty_block2 が先頭空きブロックの物理ブロック番号を示し、以降、片方向リストで空きブロックをリンクしています。空きブロックが存在しない場合、empty_block2 は 0xffffffff で、空きブロックリンクの最終ブロックのリンク先も 0xffffffff になっています。リンク先は、物理ブロック番号で示しています。ちなみに、PDIC の最適化処理というのは、空きブロックを無くし、データブロックを単語順に並び換える処理を行っています。

使用ブロックの構成には大きく分けて２種類有ります。見出語と訳語のみでよい場合と、それ以外の用例、発音記号などを必要とする場合があります。便宜上、前者を基本構成、後者を拡張構成と呼びますが、拡張構成は基本構成に拡張部を付け加えた形になっています。基本構成と拡張構成の区別はこれから述べる見出語属性のフラグによって判断します。

4.6.1 空きブロック

NULL	空きブロックリンク情報
2ﾊﾞｲﾄ	4ﾊﾞｲﾄ
空きブロックであることを示す 0x0000 が入っている	次の空きブロックへの物理ブロック番号

空きブロックはヘッダー部の empty_block2 の示すブロックから、空きブロックリンク情報をもとに片方向リンクしています。最終空きブロックの空きブロックリンク情報は 0xffffffff です。

空きブロックは連結ブロックのある使用ブロックとは異なり、必ず１ブロック単位でリンクされて構成されています。(PDIC の場合 256 バイトが 1 ブロック)

リンクは物理的にデータブロック部の先頭から後ろに順番に並んでいる必要はありません。(たとえば、空きブロック番号 100 の次のリンクが 30 であってもかまわない)

インデックス部の拡張に伴う、先頭データブロックの移動のときには、empty_block2 も変わることに注意してください。

PDIC では、使用ブロックを空きブロックにする際、先頭の６バイトを変更する以外は、NULL パディングなどをしません。これは、空きブロックを使用ブロックに復活させることを考えてのことですが、実際は利用していません。つまり、バイナリエディターで空きブロックを見てもきれいに NULL パディングされていません。

4.6.2 基本構成

使用ブロック数	フィールドデータ(複数)	フィールドデータ	終端
2ﾊﾞｲﾄ	可変長		2バイト or 4ﾊﾞｲﾄ
０の場合は空きブロックであることを示す。 最上位ビットはフィールド内のフィールド長のサイズを示す	１つのフィールドに１つの見出語が登録される		データブロックの最後を NULL2 ﾊﾞｲﾄ or 4 バイト以上で示す。 2バイトか4バイトかはフィールド長のサイズによる

PDIC の場合、最小ブロック単位は 256 バイトですが、それを複数連結することによって、大きいデータを登録できるようにしています。使用ブロック数というのはこの連結の個数を示しています。

使用ブロック数は、最大 32,767 個であるため、32,767×256=8M ﾊﾞｲﾄまで。ただし、最上位ビットはこれから述べるフィールド内のフィールド長のサイズを示します。最上位ビットが 0 の場合は、フィールド長は 2 ﾊﾞｲﾄであり、最上位ビットが 1 の場合は、フィールド長が 4 ﾊﾞｲﾄであることを示しています。

フィールド長が４バイトである場合は、１つのインデックスに対して１つのデータを割り当てなければなりません※1。つまり、１つのデータブロックには１つのデータしか登録できません。効率を考え、64K バイト以下のデータは２バイトのフィールド長を、64K バイト以上のデータは 4 バイトのフィールド長をそれぞれ使用するようにします。また、フィールド長が４バイトの場合、それに合わせてこれから述べるリンクデータのデータ長を表わすビット数も２バイトから４バイトに変わることも注意してください。

※1　4 バイト時のこの仕様は、プログラムを簡略化するためのものです。

基本構成

名称	サイズ	備考
ﾌｨｰﾙﾄﾞ長	2ﾊﾞｲﾄ or 4ﾊﾞｲﾄ	見出語から訳語の最後までのﾊﾞｲﾄ単位の長さ(ﾌｨｰﾙﾄﾞ長、圧縮長は含めない) ここのサイズはデータブロックの先頭の最上位ビットによって決まる。
圧縮長	1ﾊﾞｲﾄ	見出語部の圧縮している長さ(文字単位)を示す
見出語	可変長 NULL 終端	圧縮してある

名称	サイズ	備考
見出語属性	1ﾊﾞｲﾄ	後述
訳語	可変長 NULL 終端無し	

見出語部は圧縮されています。圧縮は同一のﾃﾞｰﾀﾌﾞﾛｯｸにおいて、直前の見出語との差分を取ることによって行っています。例えば、ﾃﾞｰﾀﾌﾞﾛｯｸの先頭の見出語が ABC であり、次の見出語が ABDEF だった場合は、２番目の見出語部に入る文字列は ABDEF ではなく、DEF となり、圧縮長は２となります。さらに ABDG という見出語が続いた場合は、圧縮長が３となり、見出語部に入る文字列は G となります。
ﾃﾞｰﾀﾌﾞﾛｯｸ先頭の見出語の圧縮長は必ず零になります。
注意：ツリービュー用の辞書では、圧縮長は０、見出語は NULL 終端のみとなっています。

見出語圧縮の例

	見出語	圧縮長	圧縮後の見出語
1 番目	ABC	0	ABC
2 番目	ABDEF	2	DEF
3 番目	ABDGD	3	GD
4 番目	AGDE	1	GDE

見出語の圧縮は一見簡単そうですが、登録・修正・削除の処理のときはかなり厄介です。圧縮文字数の性質を良く考えないと、単語が壊れたり、効率の悪いプログラムができてしまいます。圧縮無しで作りはじめ、最後に圧縮のプログラムを追加するというのも手ですが、プログラムの仕様を考える段階では必ず考慮に入れた方が楽です。(memmove は１回ですませるように)

見出語属性

次のフラグの組み合わせです。

数値	意味
0x80	必ずつける
下位 4bit	単語レベル値(0-15)
0x10	拡張構成フラグ
0x20	暗記必須単語フラグ
0x40	修正単語フラグ

0x10 の拡張構成がある場合は、次節の拡張構成になります。

4.6.3 拡張構成

名称	サイズ	備考
ﾌｨｰﾙﾄﾞ長	2ﾊﾞｲﾄ or 4ﾊﾞｲﾄ	見出語から拡張終了までのﾊﾞｲﾄ単位の長さ(ﾌｨｰﾙﾄﾞ長､圧縮長は含めない)ここのサイズはデータブロックの先頭の最上位ビットによって決まる。
圧縮長	1ﾊﾞｲﾄ	見出語部の圧縮している長さを示す
見出語	可変長 NULL 終端	
見出語属性	1ﾊﾞｲﾄ	前述
訳語	可変長 NULL 終端**あり**	
拡張属性 1	1ﾊﾞｲﾄ	後述
拡張内容 1	可変長	拡張属性によって内容は異なる
拡張属性 2	1ﾊﾞｲﾄ	
拡張内容 2	可変長	
拡張終了	1ﾊﾞｲﾄ	0x80

拡張部の特徴としては、バイナリデータ、データの圧縮などが可能になっています。また、拡張属性にはまだ未使用のフラグが残っているため､今後更に拡張される可能性があります。

拡張属性

数値	
0x01	用例
0x02	発音記号
0x03	未定義
0x04	ﾘﾝｸﾃﾞｰﾀ
0x05～0x0f	未定義
0x10	バイナリデータフラグ
0x20	未定義
0x40	圧縮フラグ
0x80	拡張終了

拡張属性は下位 4bit が項目の種類、上位 4bit が何らかのフラグであることを示します。
フラグによってさらに、拡張部の構成は異なります。
拡張部の構成は複雑です。もし、すべての拡張部を必要としない場合は、0x10 のフラグだけを見て、スキップすることができます。つまり、0x10 がなければ NULL 終端テキストであると判断し、0x10 がある場合は指定されたサイズだけスキップすれば済みます。

拡張部の構成１-バイナリデータフラグ、圧縮フラグ、拡張終了フラグがない場合

この場合、NULL 終端のあるテキストデータであると判断します。

拡張属性	1ﾊﾞｲﾄ	0x00～0x0f
テキストデータ	NULL 終端有り	

圧縮されていない用例や、発音記号はこのフォーマットです。

拡張部の構成２-バイナリデータフラグのみがある場合

拡張属性	1ﾊﾞｲﾄ	0x10～0x1f
サイズ	2ﾊﾞｲﾄ or 4ﾊﾞｲﾄ	拡張データの長さをﾊﾞｲﾄ単位で示す
拡張データ	可変長	

圧縮データや、ﾘﾝｸﾃﾞｰﾀがこのフォーマットになります。

拡張部の構成３-圧縮フラグありの場合

拡張属性	1ﾊﾞｲﾄ	0x50～0x5f
サイズ	2ﾊﾞｲﾄ or 4ﾊﾞｲﾄ	非圧縮長から圧縮データの最後までのﾊﾞｲﾄ単位の長さを示す
非圧縮長	1ﾊﾞｲﾄ	非圧縮データのﾊﾞｲﾄ単位の長さ
非圧縮データ	可変長	
圧縮ヘッダー	可変長	
圧縮データ	可変長	

圧縮ありでは、必ずバイナリデータフラグが必要です。したがって、0x50～という拡張属性になります。
非圧縮長は 1ﾊﾞｲﾄであらわすため、255ﾊﾞｲﾄを超える長さにはできないことに注意してください。
圧縮ヘッダーにはリンクデータの一部の内容が入ることがあります。(拡大率、アスペクト比など)
圧縮内容のフォーマットは省略

4.6.4 ﾘﾝｸﾃﾞｰﾀの構成

拡張属性にﾘﾝｸﾃﾞｰﾀがある場合はさらにフォーマットが定められています。

ﾘﾝｸﾃﾞｰﾀの構成 1-非圧縮時

拡張属性	1ﾊﾞｲﾄ	0x14
サイズ	2ﾊﾞｲﾄ or 4ﾊﾞｲﾄ	
リンクタイプ	1ﾊﾞｲﾄ	
ID	4ﾊﾞｲﾄ	
タイトル	可変長 NULL 終端有り	
データ部	可変長	

リンクタイプ

数値	意味	
0x00	未定義	
0x01	OLE	
0x02	ファイルリンク	
0x03	音声データ	
0x04	DDB(暫定)	
0x05	DIB(暫定)	
0x06	RTF(暫定)	
0x07～0xff	未定義	

ﾘﾝｸﾃﾞｰﾀの構成 2-圧縮時

拡張属性	1ﾊﾞｲﾄ	0x14
サイズ	2ﾊﾞｲﾄ or 4ﾊﾞｲﾄ	
非圧縮長	1ﾊﾞｲﾄ	リンクタイプからタイトルの最後までのﾊﾞｲﾄ単位の長さ
リンクタイプ	1ﾊﾞｲﾄ	
ID	4ﾊﾞｲﾄ	
タイトル	可変長 NULL 終端有り	
圧縮ヘッダー	可変長	
データ部(圧縮データ)	可変長	

さらにさらに、ﾘﾝｸﾃﾞｰﾀのデータ部はリンクタイプによって異なります。

ここではデータ部のみを示します。

ﾘﾝｸﾃﾞｰﾀの構成 1 -OLE データの場合

省略

ﾘﾝｸﾃﾞｰﾀの構成２-ファイルリンクデータの場合

省略

例：見出語、訳語、用例がある場合

名称	サイズ	備考
ﾌｨｰﾙﾄﾞ長	2ﾊﾞｲﾄ or 4ﾊﾞｲﾄ	見出語から訳語の最後までのﾊﾞｲﾄ単位の長さ
見出語	可変長 NULL 終端	
見出語属性	1ﾊﾞｲﾄ	
訳語	可変長 NULL 終端無	
拡張属性(用例)	1ﾊﾞｲﾄ	
用例	可変長 NULL 終端	
拡張終了		

4.7 PDIC におけるデータブロックの分割方法

ここでは、辞書の制御で一番難しい、データブロックの分割方法について述べます。この手のデータベースを扱ったことがない方にはピンと来ないかもしれませんが、この処理は登録の速度や格納率に大きく影響します。

辞書ファイルのサイズを抑えるためには、データブロックいっぱいにデータを詰め込み、インデックス部をなるべく小さくすることにあります。

辞書に見出語を登録する際に、見出語が昇順で順番に登録される場合は、かなり高い格納率でデータブロックを埋めていくことができますが、下手をすると未使用領域が大きくなってしまいます。NEWDIC2 までは、この点に問題があり、色んな圧縮を用いて辞書のサイズを小さくしても、生のテキストデータより若干大きくなっていました。

また、逆にデータブロックを大きくしすぎると、検索速度が落ちてしまいます。速度とサイズのトレードオフが重要です。

4.7.1 データブロックの分割の基準 - 開発者メモ

この基準を決めるには、検索速度と辞書サイズとのバランスを考える必要があります。

辞書の検索速度はインデックス部とデータ部の検索で決まります。インデックス部は B-tree で検索、データブロック部は逐次検索であるので、$O(\log Ni * \overline{Nd})$のオーダで検索されます(PDIC では、インデックス部がメモリ上、データ部がディスク上にあるのでこんなに簡単ではないのですが)。ここで、Ni はインデックス要素の個数、$\overline{Nd}$ は１つのデータブロックに入っている見出語の個数の平均、log の基底は 2 です。$\log Ni$ は小さいので、$\overline{Nd}$ によって、検索速度、辞書サイズに影響します。(Ni は $\overline{Nd}$ に依存する)

また、辞書サイズ T は次のような式で求められます。

$T ? \overline{S} ? Ni ? \overline{Li} ? Ni$

$\overline{S}$ は１つのデータブロックのサイズの平均、$\overline{Li}$ はインデックスの要素の平均長です。ちなみにデータブロックの平均格納率 $\overline{R}$ は

$$\overline{R} ? \frac{\overline{Nd} ? \overline{Ld}}{\overline{S} ? Ni}$$

です。$\overline{Ld}$ は、１つの単語の平均長です(単語といっても、訳語などすべてのデータを含め、見出語圧縮や属性、フィールド長なども含めています)。

ここで、整理してみますと、

・ $\overline{Nd}$ は小さいほうが検索が速い

・ $\overline{R}$ は大きいほうが辞書サイズが小さくなる

$\overline{R}$ は処理の結果得られるものと考えると(分割の基準というより、分割時の処理の仕方といえるので)、$\overline{Nd}$ の制御が非常に重要となります。

次に、この $\overline{Nd}$ は小さいほうがよいわけですが、小さすぎるとインデックス部が大きくなってしまい、辞書の使用効率の低下、メモリの使用量の増加、下手をするとインデックス部がいっぱいになったり、インデックス部だけでディスクが満杯になる可能性があります。また、この $\overline{Nd}$ は常に一定値ではなく、辞書の大きさ(登録単語数)に依存したほうが良いと考えられます。それは、$\log Ni$ と $\overline{Nd}$ との関係を一定にするためです。つまり、

$\log Ni ? ? \overline{Nd}$　　(αは一定値)

という関係式で考え、登録単語数を Nr とすると、

$$Nr ? \overline{Nd} ? Ni$$
$$? \overline{Nd} ? 2^{? \overline{Nd}}$$

となるように分割を制御します。

なお、αの値は実際のプログラミングに依存するので経験に依存します。

もう一つここで考慮しなければならないのは、Nr が一定値ではなく、変動値であるという点です。しかし、これはオーダレベルの話ではないので、αに含まれるものとして考えても問題ないと思います。(まてよ、αに小数点は使えないなぁ…(^^;)

次に、単語の長さ Ld を無視してきましたが、この項は理論式には現れないものの、実際には検索速度に影響を与えます。つまり、Ld が非常に大きくなると、ディスクへのアクセス時間(など)が長くなり、これがミスヒットした場合のペナルティが大きくなるためです。しかし、これもプログラムに依存する値なので、詳しくは述べません。

結論としては、

・１つのデータブロックの長さが $\overline{Nd}$ 未満であったらブロック拡張

・１つのデータブロックの長さが $\overline{Nd}$ を超えたらブロック分割

とします。ただし、$\overline{Nd}$ 未満であっても、データブロックの長さが Ld に関係した値を超えた場合も分割するとします。逆に、$\overline{Nd}$ 以上であっても、最小ブロック単位以下であったら分割しません(分割すると格納率が低下するため)。

ちなみに、PDIC ではαを 1/16(かな？)にしています。この値はブロックを分割したときの理想値にしなければならないことに注意しなければなりません。

4.7.2 データブロックの分割方法

前項で分割の基準が決まったので、次は分割方法です。ここで重要なのはデータブロックの平均格納率です。格納率を向上させるために、データブロック分割点は格納率が最も良くなるところにすればよいことになります。

これで実際のプログラミングには入れるわけですが、このブロック分割は非常に厄介な処理です。もし、登録プログラムを作成される方は、以上の内容を十分理解し、どのような変更を行ったら辞書のどの部分に影響を与えるのか、よく考えながらプログラミングされてください。まず、プログラムの仕様作成で普段の２倍の時間をかけ、一点の曇りのないものに仕上げ、それから普段の２倍の時間をかけて慎重にプログラミングを行ってください。それでも、通常のプログラムの数倍のバグが出てきます。(作者もここをしっかり押さえずにやっていたら、デバッグに恐ろしいほどの時間がかかりました(^^;;)。とにかく、いろんな条件や場合分けが存在しています。

5　NEWDIC3(Hyper 辞書形式) Ver.5.00 の構造

Ver.4.00 のヘッダー部は、アライメントを正しく取っていなかったため、それを改善したものです。Ver.4.00 との違いはヘッダー部のみです。

現在 Ver.4.00 の辞書しか出回っていませんが（2003 年 4 月現在）、徐々に Ver.5.00 へ移行していきます。互換性を考慮しなくて良い場合は Ver.5.00 で作成するようにしてください。

PDIC 側の対応予定としては、

2002 年 7 月 PDIC for Win32 Ver.4.50 以降で Ver.5.00 辞書の read/write 対応

2004 年～　PDIC for Win32 の新規作成は Ver.5.00 辞書へ。(PDIC for Win32 Ver.4.49 以前では使用不可)

Unicode 版はすべて Ver.5.00、他 OS 用で PDIC for Win32 用の辞書と互換性を考えない場合(ShiftJIS で無い場合など)は、Ver.5.00。

5.1　ヘッダー部

ヘッダー部は 256 ﾊﾞｲﾄの固定長

注意：ここで述べているデータ部のブロック番号はすべて物理ブロック番号を示しています。（物理ブロック番号についてはインデックス部を参照）

```
#define   L_HEADERNAME   100        // ヘッダー部文字列長
#define   L_DICTITLE          40          // 辞書タイトル名長

struct HEADER {
     char headername[ L_HEADERNAME ]; // 辞書ヘッダータイトル
     char dictitle[ L_DICTITLE ];      // 辞書名
     short version;                // 辞書のバージョン
     short lword;                  // 見出語の最大長
     short ljapa;                  // 訳語の最大長
     short block_size;             // １ブロックのバイト数(256 固定)
     short index_block;            // インデックスブロック数
     short header_size;            // ヘッダーのバイト数
     unsigned short index_size;    // インデックスのバイト数(未使用)

     short empty_block;            // 空きブロックの先頭物理ブロック番号（ないときは-1）
     short nindex;                 // インデックスの要素の数(未使用)
     short nblock;                 // 使用データブロック数(未使用)
     unsigned long nword;          // 登録単語数

     byte dicorder;           // 辞書の順番
     byte dictype;                 // 辞書の種別

     byte attrlen;                 // 単語属性の長さ
     byte os;                      // OS
     long olenumber;               // OLE 用シリアル番号
     ushort lid_word;              // 見出語言語 ID
```

```
    ushort lid_japa;            // 訳語部言語 ID
    ushort lid_exp;             // 用例部言語 ID
    ushort lid_pron;            // 発音記号言語 ID
    ushort lid_other;           // その他言語 ID
    byte index_blkbit;      // 0:16bit, 1:32bit
    byte dummy0;
    ulong extheader;            // 拡張ヘッダーサイズ
    long empty_block2;          // 空きブロック先頭物理ブロック番号
    long nindex2;               // インデックス要素の数
    long nblock2;               // 使用データブロック数
    byte reserved[8];
    ulong update_count;         // 辞書更新回数
    byte dummy00[4];
    byte dicident[8];           // 辞書識別子
    char dummy[ 32 ];           // 宇宙のバランスを取るためのパディング
};
```

名前	型	説明
headername	char *100	ヘッダー文字列。とくに意味はないが、以下のような文字列が必ず入る。 PDIC 自体はまったく見ていないので、Copyright の表示に使用するもよし、メッセージを書き込むもよし。 ¥x1b[2J¥x1b[32;7m ========== Dictionary for PDIC ========== ¥x1b[37;0m¥x1a
dictitle	char *40	辞書名。現在は未使用で NULL でパディング いずれ PDIC 側からどんな辞書であるか一目で分かるようにするつもりではいるが・・・。
version	short	辞書のバージョン。上位ﾊﾞｲﾄはメジャー番号、下位ﾊﾞｲﾄはマイナー番号 0x0500
lword	short	見出語の最大長。現在未参照。 248
ljapa	short	訳語の最大長。現在未参照。 3000
block_size	short	データブロックの１ブロックの長さを示す。 256 に固定 ※ここの値を変更すれば１見出語あたりの登録データを大きくすることが可能です。(PDIC はそうなるように作っていますが、動作未確認)
index_block	short	インデックスのサイズをブロック数であらわしたもの。

		１ブロックは 256 バイト インデックス部のサイズは、index_block * block_size で求めること。 このフィールドは NEWDIC,NEWDIC2 と名前が同じでも扱いがまったく異なるので注意が必要。
header_size	short	ヘッダーのサイズ。256 固定
index_size	short	インデックス部のサイズ。 NEWDIC3 では参照してはいけない！！ index_block * block_size で求めること！！
empty_block	short	先頭の空きデータブロック番号 NEWDIC3 では empty_block2 を使用すること！！
Nindex	short	インデックスの要素数(要素については後述) NEWDIC3 では nindex2 を使用すること！！
Nblock	short	使用データブロック数 NEWDIC3 では nblock2 を使用すること！！
Nword	ulong	登録単語数
Dicorder	byte	見出語の登録順 現在の PDIC はコード順のみをサポート
		0 : コード順
		1 : 大文字・小文字同一視順
		2 : 辞書順 拡張ヘッダー部に CodeMap
		3 : 降順
Dictype	byte	辞書の属性 以下のフラグの組み合わせ
		0x01 : AR 方式の圧縮を利用する
		0x08 : BOCU-1 方式の圧縮を利用(Unicode 使用時のみ)
		0x10 : Unicode(UTF-16)を使用している
		0x20 : 多国語辞書（いずれ削除）
		0x40 : 辞書をオープンするにはパスワードが必要 (辞書データそのものを暗号化するわけではない)
		0x80 : ツリービューモード用の辞書
Attrlen	byte	属性長(１に固定)
Olenumber	long	最新の OLE ｵﾌﾞｼﾞｪｸﾄ番号
Os	byte	使用 OS（この辞書が使用している文字コード体系） 以下のいずれかの値
		0x00 : DOS,Windows,OS/2(x86 系) - SJIS，改行は CR,LF
		0x01 : Mac
		0x02 : UNIX – SJIS 系
		0x03 : UNIX – EUC 系
		0x04 : UNIX – JIS 系
		0x10 : UTF-8 系
lid_word	ushort	見出語言語 ID 現在未使用
lid_japa	ushort	訳語言語 ID 現在未使用
lid_exp	ushort	用例言語 ID 現在未使用
lid_pron	ushort	発音記号言語 ID 現在未使用
lid_other	ushort	その他言語 ID 現在未使用

Extheader	ulong	拡張ヘッダーサイズ(バイト単位)
empty_block2	long	先頭空きブロック番号 ない場合は-1(0xFFFFFFFF)
nindex2	ulong	インデックス要素の数
nblock2	ulong	使用データブロック数
index_blkbit	byte	インデックス部の物理ブロック番号のビット数(0:16bit,1:32bit)
cypt	8byte	暗号コード
update_count	ulong	辞書更新回数
dummy00	4byte	予約領域
dicident	8byte	辞書識別子（ランダムな８バイトの数値）
dummy	32byte	ヘッダー部パディング

5.1　辞書識別子

NEWDIC3 Version.5 で追加された新しいヘッダー項目です。ランダムな８バイト（６４ビット）の数値が格納されます。この識別子は辞書新規作成時に与えられ、それ以降は原則として変更されません。この識別子は、ファイル名が変更されても同一の辞書であるかどうかの判断を必要とする場合に利用します。ただし、ファイルコピーで辞書を複製した場合は同一の識別子となるため、完全な識別方法として利用することはできません。

※プログラミング上の注意※

乱数の発生方法に注意してください。例えば、

```
srand(time(NULL)); for(i=0;i<8;i++)dicident[i]=rand();
```

ですと、srand()だけで識別子が決まることになり、同じ時刻に作成した辞書は必ず同じ識別子になってしまいます。rand 関数は乱数のように見えるだけであり、本当の乱数ではありません。

5.2　辞書順辞書（予告）

Ver.5 より辞書順辞書に正式対応します。

辞書順辞書とは、大文字・小文字を同一視たり、a と a を同一視、ハイフンを無視、といった紙の辞書と同様な語順にする辞書です。

6　Unicode(BOCU) 辞書 (Hyper 辞書形式 Ver.5)

Unicode に対応した辞書形式です。文字コードには BOCU 方式の圧縮を行った Unicode を用います。BOCU は Binary-Ordered Compression for Unicode の略で、Unicode に向いた簡単で便利な圧縮方法です。

（BOCU の詳細については http://www-6.ibm.com/jp/developerworks/unicode/010921/j_u-binary.html などを参考）

BOCU の特徴は、参考 URL に記載されている通り、

・欧米文字はほぼ１バイト（UTF-8 とほぼ同等）

・日本語ではほぼ２バイト（UTF-8 なら３バイト）

・バイナリ―オーダーが保存される（文字列ソートにおける前後関係を圧縮したまま調べることができる）

という特徴があり、辞書へ保存するには適した形式です。

さらに、

・UTF-8 や UTF-16 といったどのエンコーディングにも同等に対応できる(＝CodePoint そのものを扱っている)

・エンディアンに依存しない

・文字列終端はシングルバイトの'¥0'で表す

という特徴もあります。

欠点としては、圧縮伸長処理が必要になります。しかし BOCU は単純な差分による圧縮方式であるため、ディスクアクセスに比べれば小さいオーバーヘッドで済み、パフォーマンスに大きな差はないはずです。

6.1　ShiftJIS コード辞書との違い

・文字コードはすべて BOCU1 encoding

・各項目の最大バイト数の制限を緩和（カッコ内は ShiftJIS 時）

- 見出語 248（248）バイト
- 訳語 30,000 (3,000) バイト
- 用例 100,000 (20,000) バイト
- 発音記号 1,000 (1,000) バイト

・格納できる文字数は圧縮後のサイズに依存するため不定

※PDIC では見出語部以外のサイズを厳密に規定していません。このサイズを越えて辞書に登録する場合があります。

6.2　ヘッダー部

青字は ShiftJIS コードの辞書と異なる個所を示す。

名前	型	説明
headername	char *100	ヘッダー文字列。とくに意味はないが、以下のような文字列が必ず入る。

		“ ========== Dictionary for PDIC ========== “(ASCII encoding)
dictitle	char *40	辞書名。現在は未使用で NULL でパディング（BOCU1 encoding を使用）
version	short	辞書のバージョン。上位ﾊﾞｲﾄはメジャー番号、下位ﾊﾞｲﾄはマイナー番号 0x0500
lword	short	見出語の最大長。現在未参照。 248
ljapa	short	訳語の最大長。現在未参照。 30,000
block_size	short	データブロックの１ブロックの長さを示す。 256 に固定
index_block	short	インデックスのサイズをブロック数であらわしたもの。 １ブロックは 256 バイト インデックス部のサイズは、index_block * block_size で求めること。 このフィールドは NEWDIC,NEWDIC2 と名前が同じでも扱いがまったく異なるので注意が必要。
header_size	short	ヘッダーのサイズ。256 固定
index_size	short	インデックス部のサイズ。 NEWDIC3 では参照してはいけない！！ index_block * block_size で求めること！！
empty_block	short	先頭の空きデータブロック番号 NEWDIC3 では empty_block2 を使用すること！！
nindex	short	インデックスの要素数(要素については後述) NEWDIC3 では nindex2 を使用すること！！
nblock	short	使用データブロック数 NEWDIC3 では nblock2 を使用すること！！
nword	ulong	登録単語数
dicorder	byte	見出語の登録順 現在の PDIC はコード順のみをサポート
		0 コード順
		1 大文字・小文字同一視順
		2 辞書順
		3 降順
dictype	byte	辞書の属性 以下のフラグの組み合わせ
		0x01 バイナリの圧縮を行っている
		0x08 BOCU1 方式の圧縮を利用
		0x10 Unicode を使用している

		0x20	多国語辞書（いずれ削除）
		0x40	辞書をオープンするにはパスワードが必要 (辞書データそのものを暗号化するわけではない)
		0x80	ツリービューモード用の辞書
attrlen	byte	属性長(１に固定)	
os	byte	使用 OS Unicode は必ず 0x00	
		0x00	DOS,Windows,OS/2(x86 系), 改行は CR,LF
		0x01	Mac
		0x02	UNIX - SJIS 系
		0x03	UNIX – EUC 系
		0x04	UNIX - JIS 系
		0x20	BOCU encoding
olenumber	long	最新の OLE ｵﾌﾞｼﾞｪｸﾄ番号	
lid_word	ushort	見出語言語 ID 現在未使用	
lid_japa	ushort	訳語言語 ID 現在未使用	
lid_exp	ushort	用例言語 ID 現在未使用	
lid_pron	ushort	発音記号言語 ID 現在未使用	
lid_other	ushort	その他言語 ID 現在未使用	
index_blkbit	byte	インデックス部の物理ブロック番号のビット数(0:16bit,1:32bit)	
dummy0	byte		
extheader	ulong	拡張ヘッダーサイズ(バイト単位)	
empty_block2	long	先頭空きブロック番号 ない場合は-1(0xFFFFFFFF)	
nindex2	ulong	インデックス要素の数	
nblock2	ulong	使用データブロック数	
cypt	8byte	暗号コード	
update_count	ulong	辞書更新回数（LAN 共有時に使用）	
dummy00	4byte	予約領域	
dicident	8byte	辞書識別子（ランダムな８バイトの数値）	
dummy	32byte	ヘッダー部パディング	

※DicType の Unicode(UTF-16)用辞書ビットは立ちません

※os と olenumber が入れ替わり、index_blkbit の位置も異なります。

データ部：

名称	サイズ	備考
ﾌｨｰﾙﾄﾞ長	2ﾊﾞｲﾄ	見出語から訳語の最後までのﾊﾞｲﾄ単位の長さ(ﾌｨｰﾙﾄﾞ長、見出語属性、圧縮長は含めない) ここのサイズはデータブロックの先頭の最上位ビットによって決まる。
圧縮長	1ﾊﾞｲﾄ	見出語部の圧縮している長さ(バイト単位)を示す
見出語属性	1ﾊﾞｲﾄ	
見出語	可変長 NULL 終端	圧縮長を除いた残りの見出語部分

名称	サイズ	備考
		BOCU1 encoding
訳語	可変長 NULL 終端無し	BOCU1 encoding

(1)訳語部、用例部、オブジェクトタイトルなどテキストを使用する部分はすべてを BOCU encoding を用いる。

(2)見出語属性の位置が異なる（見出語属性が見出語圧縮長の直後に位置している）

6.3 BOCU1 圧縮方式

BOCU(Binary-Order Compression for Unicode)1 は名前のとおり、Unicode 用の文字列圧縮方式です。

http://oss.software.ibm.com/cvs/icu/~checkout~/icuhtml/design/conversion/bocu1/bocu1.html

7　Unicode(UTF-8) 辞書

※この形式の辞書は試作における一時的なバージョンのための形式であり、いずれ削除されます。

現在のところ、UTF-8 が使用できる辞書は Ver.5 のみです。

ここでは Ver.5 を ShiftJIS で使用した場合と異なる点を示しておきます。

7.1　ShiftJIS 用の辞書と異なる点

ヘッダー部：

青字は ShiftJIS 辞書との違う個所を示す。

名前	型	説明
headername	char *100	ヘッダー文字列。とくに意味はないが、以下のような文字列が必ず入る。 PDIC 自体はまったく見ていないので、Copyright の表示に使用するもよし、メッセージを書き込むもよし。 “ ========== Dictionary for PDIC ========== “
dictitle	char *40	辞書名。現在は未使用で NULL でパディング（UTF-8 を使用）
version	short	辞書のバージョン。上位ﾊﾞｲﾄはメジャー番号、下位ﾊﾞｲﾄはマイナー番号 0x0500
lword	short	見出語の最大長。現在未参照。 248
ljapa	short	訳語の最大長。現在未参照。 3000
block_size	short	データブロックの１ブロックの長さを示す。 256 に固定
index_block	short	インデックスのサイズをブロック数であらわしたもの。 １ブロックは 256 バイト インデックス部のサイズは、index_block * block_size で求めること。 このフィールドは NEWDIC,NEWDIC2 と名前が同じでも扱いがまったく異なるので注意が必要。
header_size	short	ヘッダーのサイズ。256 固定
index_size	short	インデックス部のサイズ。 NEWDIC3 では参照してはいけない！！ index_block * block_size で求めること！！
empty_block	short	先頭の空きデータブロック番号 NEWDIC3 では empty_block2 を使用すること！！

nindex	short	インデックスの要素数(要素については後述) NEWDIC3 では nindex2 を使用すること！！	
nblock	short	使用データブロック数 NEWDIC3 では nblock2 を使用すること！！	
nword	ulong	登録単語数	
dicorder	byte	見出語の登録順 現在の PDIC はコード順のみをサポート	
		0	コード順
		1	大文字・小文字同一視順
		2	辞書順
		3	降順
dictype	byte	辞書の属性 以下のフラグの組み合わせ	
		0x01	AR 方式の圧縮を利用
		0x08	BOCU-1 方式の圧縮を利用
		0x10	Unicode を使用している
		0x20	多国語辞書（いずれ削除）
		0x40	辞書をオープンするにはパスワードが必要 (辞書データそのものを暗号化するわけではない)
		0x80	ツリービューモード用の辞書
attrlen	byte	属性長(１に固定)	
os	byte	使用 OS Unicode は必ず 0x00	
		0x00	DOS,Windows,OS/2(x86 系), 改行は CR,LF
		0x01	Mac
		0x02	UNIX - SJIS 系
		0x03	UNIX – EUC 系
		0x04	UNIX - JIS 系
		0x10	UTF-8
olenumber	long	最新の OLE ｵﾌﾞｼﾞｪｸﾄ番号	
lid_word	ushort	見出語言語 ID 現在未使用	
lid_japa	ushort	訳語言語 ID 現在未使用	
lid_exp	ushort	用例言語 ID 現在未使用	
lid_pron	ushort	発音記号言語 ID 現在未使用	
lid_other	ushort	その他言語 ID 現在未使用	
index_blkbit	byte	インデックス部の物理ブロック番号のビット数(0:16bit,1:32bit)	
dummy0	byte		
extheader	ulong	拡張ヘッダーサイズ(バイト単位)	
empty_block2	long	先頭空きブロック番号 ない場合は-1(0xFFFFFFFF)	
nindex2	ulong	インデックス要素の数	
nblock2	ulong	使用データブロック数	
cypt	8byte	暗号コード	
update_count	ulong	辞書更新回数（LAN 共有時に使用）	
dummy00	4byte	予約領域	
dicident	8byte	辞書識別子（ランダムな８バイトの数値）	

dummy	32byte	ヘッダー部パディング

※DicType の Unicode(UTF-16)用辞書ビットは立ちません

※os と olenumber が入れ替わり、index_blkbit の位置も異なります。

データ部：

名称	サイズ	備考
ﾌｨｰﾙﾄﾞ長	2ﾊﾞｲﾄ	見出語から訳語の最後までのﾊﾞｲﾄ単位の長さ(ﾌｨｰﾙﾄﾞ長、見出語属性、圧縮長は含めない) ここのサイズはデータブロックの先頭の最上位ビットによって決まる。
圧縮長	1ﾊﾞｲﾄ	見出語部の圧縮している長さ(**バイト単位**)を示す
見出語属性	1ﾊﾞｲﾄ	
見出語	可変長 NULL 終端	圧縮長を除いた残りの見出語部分
訳語	可変長 NULL 終端**無し**	BOCU-1 で圧縮

(3)見出語部は UTF-8 のコードをそのまま使用する（BOCU-1 による圧縮は行わない）。

(4)訳語部、用例部、オブジェクトタイトルなどテキストを使用する部分はすべてを BOCU-1(Binary-Order Compression for Unicode)による圧縮を行う。(BOM は付加しない)

(5)フィールド部が異なる（見出語属性が見出語圧縮長の直後に位置している）

7.2 BOCU-1 圧縮方式

BOCU-1(Binary-Order Compression for Unicode)は名前のとおり、Unicode 用の文字列圧縮方式です。

http://oss.software.ibm.com/cvs/icu/~checkout~/icuhtml/design/conversion/bocu1/bocu1.html

※2002.6.6 時点では Draft であるためリンクが切れている可能性がある

※Not Found であった場合は、http://www.Unicode.org より、**BOCU-1** で検索してください

7.2.1 BOCU-1 の圧縮効率と検索速度

例えば ShiftJIS で書かれた英和辞書テキストファイルを Unicode へ変換し BOCU-1 圧縮した場合、だいたい同じサイズくらいになります。

ということは、ShiftJIS 辞書とは辞書サイズの大きな差はないが、BOCU-1 の伸長処理が余計にかかるため検索が遅くなります。また、BOCU-1 圧縮しなかった場合は辞書サイズが３０～４０%程度(UTF-8 の場合)増えるため、その分だけディスクアクセスが増加することになり、やはり検索速度は遅くなります。

従って、英和・和英辞書しか利用しない人にとっては、Unicode 辞書を利用することは不利ということになります。（多くのアジア人は Unicode の仕様に問題ありと声を上げるでしょう）

ただ、圧縮率に関しては ShiftJIS よりはやや小さくなる傾向にあります。それが気持ち程度の救いでしょうか。。。

8　Unicode(UTF-16)対応辞書

※この形式の辞書は試作における一時的なバージョンのための形式であり、いずれ削除されます。

Unicode(UTF-16)用の辞書形式について述べます。

UTF-16 では、英数字でも２バイト必要とすることから、辞書の構造が変更されています。PDIC 辞書でも例外無く、文字として扱う部分は、その長さをバイト数ではなく、文字数として扱います。例えば、見出語の圧縮文字数。これにより、見出語の長さ最大 248 文字分を圧縮可能になっています。したがって、見出語部は、必ず２バイト単位となり、見出語部の終端を示す NULL 文字も２バイトで 0x0000 となります。さらに、圧縮文字単位も１文字(２バイト)単位となります。

さらに Unicode 辞書の場合は WindowsCE での使用を考慮してアライメントを考えて各項目は配置されています。つまり、ワードデータがワード境界を跨ぐような配置を避けるように変更されています。

フィールド長は文字を扱うものではないので、そのままバイト単位で１単語分のデータの長さを示します。

なお、Unicode 対応辞書は、現在のところ、Hyper 辞書形式のみです。将来的にも、Hyper 辞書形式以外はサポートの予定はありません。

注意！Unicode 対応辞書の辞書バージョンは Ver.5.00 です。

8.1　Unicode 対応辞書の判別

1.ヘッダー部のオフセット 0xA5 にある、辞書種別で Unicode であるかどうか判断。0x10 が立っていれば Unicode 対応辞書。

2.ヘッダー部のオフセット 0x00 にある、ヘッダー文字列で Big-Endian(Mac)か Little-Endian(Intel)の判断を行う。0xFE 0xFF と続けば Big-Endian、0xFF 0xFE と続けば Little-Endian となる。(Unicode のファイルの byte-order mark 規定により)

8.2　Unicode 辞書と ANSI 辞書の違い

以下に各ブロックの違いを青字で示す。

ヘッダー部

名前	型	説明
Headername	wchar_t *100	¥xFE¥xFF
Dictitle	wchar_t *40	辞書名。現在は未使用で NULL でパディング
Version	short	辞書のバージョン。上位ﾊﾞｲﾄはメジャー番号、下位ﾊﾞｲﾄはマイナー番号 0x0400
Lword	short	見出語の最大長。現在未参照。 248
ljapa	short	訳語の最大長。現在未参照。 3000

block_size	short	データブロックの１ブロックの長さを示す。 256 に固定	
index_block	short	インデックスのサイズをブロック数であらわしたもの。 １ブロックは 256 バイト インデックス部のサイズは、index_block * block_size で求めること。 このフィールドは NEWDIC,NEWDIC2 と名前が同じでも扱いがまったく異なるので注意が必要。	
header_size	short	ヘッダーのサイズ。256 固定	
index_size	short	インデックス部のサイズ。 NEWDIC3 では参照してはいけない！！ index_block * block_size で求めること！！	
empty_block	short	先頭の空きデータブロック番号 NEWDIC3 では empty_block2 を使用すること！！	
nindex	short	インデックスの要素数(要素については後述) NEWDIC3 では nindex2 を使用すること！！	
nblock	short	使用データブロック数 NEWDIC3 では nblock2 を使用すること！！	
nword	ulong	登録単語数	
dicorder	byte	見出語の登録順 現在の PDIC はコード順のみをサポート	
		0	コード順
		1	大文字・小文字同一視順
		2	辞書順
		3	降順
dictype	byte	辞書の属性 以下のフラグの組み合わせ UTF-16 辞書では 0x18 または 0x10 となります。	
		0x01	AR 方式の圧縮を利用
		0x08	**BOCU-1 方式の圧縮を利用**
		0x10	**Unicode を使用している**
		0x20	多国語辞書（いずれ削除）
		0x40	辞書をオープンするにはパスワードが必要
		0x80	ツリービューモード用の辞書
attrlen	byte	属性長(１に固定)	
os	byte	使用 OS Unicode は必ず 0x00	
		0x00	DOS,Windows,OS/2(x86 系), 改行は CR,LF
		0x01	Mac
		0x02	UNIX - SJIS 系

		0x03	UNIX - EUC 系
		0x04	UNIX - JIS 系
olenumber	long	最新の OLE ｵﾌﾞｼﾞｪｸﾄ番号	
lid_word	ushort	見出語言語 ID 現在未使用	
lid_japa	ushort	訳語言語 ID 現在未使用	
lid_exp	ushort	用例言語 ID 現在未使用	
lid_pron	ushort	発音記号言語 ID 現在未使用	
lid_other	ushort	その他言語 ID 現在未使用	
index_blkbit	byte	インデックス部の物理ブロック番号のビット数(0:16bit,1:32bit)	
dummy0	byte		
extheader	ulong	拡張ヘッダーサイズ(バイト単位)	
empty_block2	long	先頭空きブロック番号 ない場合は-1(0xFFFFFFFF)	
nindex2	ulong	インデックス要素の数	
nblock2	ulong	使用データブロック数	
cypt	8byte	暗号コード	
update_count	ulong	辞書更新回数（LAN 共有時に使用）	
dummy00	4byte	予約領域	
dicident	8byte	辞書識別子（ランダムな８バイトの数値）	
dummy	32byte	ヘッダー部パディング	

インデックス部

物理ﾌﾞﾛｯｸ番号	ﾌﾞﾛｯｸ先頭の見出語		インデックス部の最後
2 バイト or ４バイト	可変長 (Unicode)		4 バイト
0 から始まる物理ブロック番号を示す	NULL 終端 (2byte)		NULL ４ﾊﾞｲﾄ以上であらわす

データ部

名称	サイズ	備考
ﾌｨｰﾙﾄﾞ長	2 ﾊﾞｲﾄ	見出語から訳語の最後までのﾊﾞｲﾄ単位の長さ(ﾌｨｰﾙﾄﾞ長、見出語属性、圧縮長は含めない) ここのサイズはデータブロックの先頭の最上位ビットによって決まる。
圧縮長	1 ﾊﾞｲﾄ	見出語部の圧縮している長さ(文字単位)を示す
見出語属性	1 ﾊﾞｲﾄ	
見出語	可変長 NULL 終端	圧縮してある

名称	サイズ	備考
	(Unicode)	
訳語	可変長 NULL 終端**無し** (Unicode)	BOCU-1 で圧縮されている場合は２バイト単位でサイズのアライメントをとる。

発音記号、用例、オブジェクトタイトルも同様に Unicode NULL 終端とする。

拡張構成

名称	サイズ	備考
ﾌｨｰﾙﾄﾞ長	2ﾊﾞｲﾄ	見出語から拡張終了までのﾊﾞｲﾄ単位の長さ(ﾌｨｰﾙﾄﾞ長、見出語属性、圧縮長は含めない) ここのサイズはデータブロックの先頭の最上位ビットによって決まる。
圧縮長	1ﾊﾞｲﾄ	見出語部の圧縮している長さを示す
見出語属性	1ﾊﾞｲﾄ	
見出語	可変長 NULL 終端	
訳語	可変長 NULL 終端**あり**	BOCU-1 で圧縮されている場合は２バイト単位でアライメントをとる。
拡張属性 1	1ﾊﾞｲﾄ	
Reserved	1ﾊﾞｲﾄ	0x00
拡張内容 1	可変長	拡張属性によって内容は異なる
拡張属性 2	1ﾊﾞｲﾄ	
Reserved	1ﾊﾞｲﾄ	0x00
拡張内容 2	可変長	
拡張終了	1ﾊﾞｲﾄ	0x80
Reserved	1ﾊﾞｲﾄ	0x00

※詳細な説明は省略

※現在のところ、フィールド長が４バイトになる場合は未定義です。

※バイナリデータの圧縮方式は ShiftJIS 辞書とは異なります。（詳細は省略）

※拡張内容がテキストである場合は、BOCU-1 による圧縮を行います(DicType に 0x08 がある場合)

8.3 BOCU-1 の圧縮された場合のアライメント方法

２バイト単位のアライメントに配置されるように NULL で最後をパディングする必要があります。

拡張構成でない場合は訳語のみなので NULL 終端はありませんが、アライメントをとるために最大１バイトの NULL を入れる必要があります。
拡張構成の場合は、訳語部・用例部・発音記号部には２～３バイトの NULL 終端になります。
例えば、拡張構成時、BOCU-1 圧縮後の最後のデータが奇数アドレスに配置された場合は、
[最後のバイト] [00] [00] [00]
というように、パディングします。（１バイトパディングでアライメントをとってはいけない）

9　履歴

1997.2.11　0.4 版 Release(Hyper 辞書形式追加)
2000.8.19　0.5 版 Release (Unicode 対応追加)
2002.3.2　0.6 版 Release(Unicode 版の圧縮方式記述)
2002.6.6　0.7 版 Release Ver.5 仕様追加、Unicode(UTF-8)追加、辞書順辞書追加、読取専用辞書追加
2003.4.28　0.8 版 Release Unicode(BOCU)版追加

10　著作権