PHI CD ROM Format Description

19 April 1992

The Packard Humanities Institute
300 Second Street, Suite 201
Los Altos, CA 94022
Telephone (415) 948-0150
bitnet xb.m07@stanford
Fax (415) 948-5793

Significant changes

The format of the data on the present CD ROM differs from that on the last PHI CD ROM primarily in the following respects:

1. A new ID level (**n**) has been added for documents. When this level is used the lower levels (**v..z**) are not hierarchical and can vary independently; furthermore the number of active levels can vary from document to document within a given work.

2. Descriptive information other than the traditional citation is now allowed in the ID data; for example, the ID may contain the date or location of a papyrus.

3. The information in the ID table is no longer sorted but is given in the same order as it appears in the text file. In order to locate a specific passage the entire ID table must be considered.

4. A different algorithm is used in comparing IDs to determine what lines in a text file are "exceptions".

5. A new field has been added to the ID table to indicate a single-line exception.

6. A new **d** level has been added to indicate the preferred abbreviated author name.

PHI CD ROM Format

The recent CD ROMs issued by the TLG and PHI follow the ISO 9660:1988 (E) standard for volume and directory structure. Please note, however, that the internal organization of the files does not conform to the optional "variable length record" structure defined in that standard.

You may be able to use a standard software driver to locate the files in the directory and read the file data from the CD ROM, but your program will need to read the file in binary mode and extract the text records from the blocks according to the format information presented in this document.

Most files on the CD ROM are either text files or ID Table files.

Each text file (designated by the filename extension .TXT) consists of variable-length text records, delimited by compressed binary-coded ID citations. The binary ID format is described below on page 5; in summary, each byte of id information has the high-order bit set. The text itself is in 7-bit ASCII coded according to the conventions described in the document "Beta Coding Summary."

Each ID Table file (designated by the filename extension .IDT) is a table of contents to the corresponding text file, designed to facilitate rapid access to particular sections of the text. The ID Table file has a complex structure, and some applications may choose not to make any use of it. A text file is fully usable without any reference to its ID Table file. Each ID Table file is guaranteed to be allocated immediately before its associated .TXT file to facilitate accessing these files by reading consecutive sectors.

The Author List (with the filename AUTHTAB.DIR) contains descriptive information for each text file on the disc. The purpose of the Author Table is to allow the user to ask for the author Plato, for example, without having to know that the actual file name is TLG0059. Each entry contains the author name, the corresponding file name, synonyms, remarks, and language. The entries are arranged by category.

Text Files

A text file usually contains the writings (encoded as necessary) of one or more ancient authors. These all carry a traditional citation system. There are other kinds of text files, though, which may contain (e.g.) bibliographic data or morphologically analyzed text. For consistency, these texts also carry a citation system (usually a simple line increment).

Text files are organized in blocks of 8192 bytes. Each block begins with the full citation for the first record of the block. Subsequent records are preceded by an abbreviated citation. Since the ID bytes are all marked with the sign bit set, the citation serves to separate the variable length text records from one another. The end of block is signalled by an end of block marker in an ID field following the last record of the block. End of file is indicated by a marker preceding the end of block marker for the final block. Records do not span blocks.

Processing a block of text is therefore simple. Read in all bytes with the sign bit set. This is the ID for the first record. Call a subroutine to decode the ID data. Now read in all bytes with the sign bit unset. This is the text of the first record. Call a subroutine to process the text. Repeat this process for all records in the block, that is, until the ID data contains the end of block marker.

For a description of the text format for Greek texts, see the document "Beta Coding Summary."

ID Data

The format of the ID data is the same for text (.TXT) and ID (.IDT) files. A single subroutine can therefore be written to decode the ID information from both file types. The data includes both strictly citational information and unstructured descriptive material; the two are independent of each other and since the descriptive information (where it exists) always follows the citation information, it may easily be disregarded by a simplified decoding subroutine. Included in the data are codes specifying the ID level (for citations, **a..d,n,v..z**; for descriptors, **a..z**) and the ID value. In addition, certain control codes (end-of-block, end-of-file) are included among the ID bytes.

Citation data

The ID levels **a** and **b** are reserved for the citation of the author and work respectively. These levels occur in every text. The **c** level is an optional level specifying the preferred abbreviation for a work (this is used where, as at the TLG and PHI, the work is cited by a control number). The **d** level also is optional and specifies the preferred abbreviation for an author (for example, at PHI, "Verg" is the **d** level for Vergil). As an example, Vergil's Aeneid will have citations at the **a** through **d** levels of **a** = "0690", **b** = "003", **c** = "A", and **d** = "Verg". The optional **c** and **d** levels are not included in the ID Table files, since each work is fully identified by the **a** and **b** levels.

The lower levels, **n** and **v** through **z**, are used to cite fields within an individual work. For a given work these behave according to one of two schemes. In the first (this is the only scheme used by the TLG and PHI prior to 1990), the **n** level is not used and levels **v** through **z** are used hierarchically: the field varying most rapidly is always **z** and denotes the line number; the other levels are used only as needed. Thus in the New Testament the **x** level is the chapter, the **y** level is the verse, and the **z** level is the line. The number of levels within a work is constant. This type of citation is typically used for literary texts.

In the second scheme, the **n** level is always present (its presence or absence alone indicates which scheme is in effect) and is used to specify a "document" within a work. Levels **v** through **z**, in this scheme, are subordinate to **n** (that is, when **n** changes **v** through **z** become null) but they are not otherwise arranged hierarchically: they change independently of one another. The **z** level is reserved for line number but the other levels, **v** through **y** are assigned to whatever fields are appropriate to the document at hand. This type of citation allows for handling the individual inscriptions or papyri within a single volume (work), each of which may have varying numbers of ID levels for information such as fragments, sides and columns.

Descriptive data

The optional descriptor ID levels (**a..z**) are used independently of levels **a..d,n,v..z** to hold comments or descriptive information. They are not part of the citation scheme and are not themselves hierarchical. The comment contained in a descriptor ID level applies to all the text lines that follow until the value of that descriptor level changes or a change in the work or document level sets all the descriptor levels to null. Their assignment (level **l**, for example, to indicate the location of a papyrus, or **d** to indicate its date) is determined by the data preparer. Although there are twenty-six possible descriptor ID levels (**a..z**), PHI has used no more than eight in a single document. PHI reserves the **z** descriptor level as a comment sequence number within a work: in the display of continuous text (with optimized ID's), it facilitates determining where the data preparer intended a comment to appear but has no other conventional meaning and is not part of the original comment. These descriptors are not included in ID Table files.

ID values

ID values are divided into binary and ASCII components. Leading digits, if any, are converted into a binary value; any trailing characters become the ASCII component. Thus, the citation "12" has a binary value of 12 and no ASCII value. The citation "12a" also has a binary value of 12 but an ASCII value of "a". The citation "a12" has no binary value and and ASCII value of "a12".

The ASCII component presently can have a total length from 1 to 15 characters (citations) or 1 to 31 characters (descriptors) as used on the CD ROMs being described; these values may change in future releases.

When citations are compared, the binary value is compared first. If the binary values match, the ASCII values are converted to lower case and compared character by character, but runs of digits within the ASCII string are evaluated as numbers. Thus the citation "3a" is less than the citation "12a" (since binary 3 is less than binary 12) and citation "a3" is less than the citation "a12" (even though ASCII 3 is greater than ASCII 1); "3B" is greater than "3a"; and "t" is less than "1". By the same rules, "A31" is less than both "A300" and "AB", since 31 comes before 300 (numeric) and "A" comes before "AB" (string).

Numbers can range from 1 to 16383; larger numbers are treated (and sorted) as strings.

An ID level is explicitly set to null if it consists of a null string, coded with no binary value followed by a string of length zero.

Decoding

An ID byte may be distinguished from a text byte by the high bit (the sign bit) of the byte. Since the text encoding system is based on 7-bit ASCII characters, the sign bit is always clear for text bytes; the sign bit is always set for ID bytes. This distinction makes it easy to separate ID information from text information as the data is processed.

The first byte of an ID sequence is always a code byte. The code byte is followed by data bytes, as required. Additional code bytes with their data bytes may follow. Descriptor code and value bytes, where they exist, always follow citation bytes.

In order to process a code byte, the left and right hand nibbles must be isolated. The left nibble will usually contain the level code and the right nibble will contain information about the ID value for that level. When processing any attendant data bytes, the sign bit must first be stripped. For ASCII data, one need only clear the sign bit. For binary data, though, it is necessary to consider the value exclusive of the sign bit. Thus a two-byte binary value contains only 14 bits of information (the lowest seven bits of each byte).

Left nibble

Since the sign bit is always set, there are eight possible values for the left nibble.

1000	**z**-level ID
1001	**y**-level ID
1010	**x**-level ID
1011	**w**-level ID
1100	**v**-level ID
1101	**n**-level ID
1110	Escape code: ID level will be found in next ID byte
1111	Special code (not an ID): see below

Right nibble

The right nibble has sixteen possible values. Since low binary values are the most common ID values, 1-7 are reserved as literal values. The ID can therefore be expressed as a single byte in many cases.

0000	increment the ID at this level
0001-0111	literal binary ID values
1000	7-bit binary value
1001	7-bit binary value + single ASCII character
1010	7-bit binary value + ASCII string
1011	14-bit binary value
1100	14-bit binary value + single ASCII character

1101	14-bit binary value + ASCII string
1110	same binary value + new single ASCII character
1111	no binary value + ASCII string

Escape codes

When the left nibble is binary 1110, the right nibble contains information on the ID value, as above. The level code is, however, contained in the next byte. This level code occupies the full byte (disregarding the sign bit) and should be processed immediately, as it will intervene between the right nibble code and any data bytes which follow.

The values defined by the escape code usually describe high level citation ID fields (ie, author, work) or descriptor ID's. The level code contained in the next byte has for citation ID's the possible values: **a**=0, **b**=1, **c**=2, and **d**=4; for descriptor ID's: **a**=97, **b**=98, **c**=99, ... , **z**=122. Descriptor ID's thus always begin with an escape code (left nibble is binary 1110) and always have a level code (sign bit disregarded) greater than 96.

Special codes

When the left nibble is an all ones value (1111), the right nibble defines a special code usually a delimiter.

1111 1111	end-of-ASCII-string
1111 1110	end-of-block
1111 0000	end-of-file
1111 1000	exception start
1111 1001	exception end

The end-of-block code is the last valid data byte in every block; the rest of the block is padded with nulls. The end-of-file code is the next-to-the-last data byte in the last block of every file: it is followed by an end-of-block code and null padding.

Exception-start and exception-end are included optionally to delimit text lines that appear out-of-order (when evaluated by the comparison technique described above). These codes are never needed to determine the current ID and may be ignored; they are intended only to serve as hints in browsing through a text rearranged from its traditional order by a modern editor.

Abbreviated ID Fields

In ID files, the full ID is given for each author ID (level **a** only), each work ID (level **b** only) and each new section (levels **n** and **v-z**); descriptor ID levels are not included in ID files. In text files, the full ID (all levels: citation and descriptor) is given at the beginning of each 8K block. Other ID fields usually contain only enough information to show how the current ID field differs from the last. Thus,

most lines in a text require only the code for "increment the **z** level" (binary 1000 0000). When the higher levels do not change, they need not be cited. When a higher level does change and levels **v** through **z** are hierarchical (that is, no **n** level is present), all lower levels are implicitly set to binary 1. This often obviates the need to cite the lower levels explicitly. Thus, to mark line 1 of Chapter 2 in a (hierarchical) work, the required citation would be "increment the **y** level" (binary 1001 0000) since the **y** level was previously set for Chapter 1 and the **z** level is set to 1 implicitly. Note that when the author or work changes, all lower levels are set to null.

When levels **v** through **z** are not hierarchical (that is, in documents, where the **n** level is present), a change in author, work or document (**n**-level) sets all lower levels to null. Otherwise the lower levels are set explicitly.

Level descriptions (e.g., **x**="book", **y**="chapter") in the ID table are handled in a similar fashion. The full description is always given for the author level, the work level, and for the lower levels of the first work. Lower level descriptions for subsequent works may be omitted if unnecessary. Thus, if thirty consecutive works are cited by Book/line, the table need only give this information for the first instance only.

Coding

On PHI CD ROMs, ASCII information in the citations is treated literally, and the Beta code conventions used in text data are not applicable. Beta code conventions **are** used to ASCII text in descriptor ids (those with level codes 96, 97, 98, ..., 122) and in the level descriptions.

ID Table Files

Description

For each text file, the corresponding ID table file provides a detailed account of the identity and location of the authors and works for that file, the location of all major sections within the works, and a complete listing of the ending citation for each text block within a section. In the case of documents, information is given to the document (**n**) level only for the end of each text block and information about lower levels is not included. If the document id is the same at the end of consecutive blocks, the first block is marked with the document id, and the later ones have the new block code without any additional information.

For example, the ID table for file TLG0012.TXT would tell us that the first author is named "Homer" and that the author is cited "0012", that the first work is named "Iliad" and is cited "001", and that the first major section of this work is is Book 1. The block location for each of these is given, and the section data is followed by a list of the ending citations for each block in Book 1. The data for Book 1 will be followed by that for Book 2, and so forth until the second work is encountered.

Note that each subdivision is nested, that is, the text for an author is divided into one or more works, the text for a work is divided into one or more sections, and the text for a section is divided into one or more blocks. A block may contain parts of two or more sections or works; work and section boundaries do not have to coincide with block boundaries. The works and sections are presented in the ID Table file in the same order as they are found in the text; they are not sorted.

Because an editor will at times reorder a text but leave the traditional citation intact, the ID table makes provision for out-of-sequence lines. If an editor places a line numbered 912 between lines 310 and 311, this will usually produce an exception field. An exception field lists the beginning and ending citation for lines which do not fall in the expected block. Note that if an editor positions line 314 between lines 310 and 311, this will not usually produce an exception. The reason for this is that line 314 is very likely in the expected block, despite the fact that it is out of order within the block. Thus, to find line 912 in the example above, you would locate the block in the usual fashion. Immediately before the block which contains, say, lines 885-940, the exception would be listed along with the true block location for the line.

Format

Each entry in the ID table is introduced by a type code byte from zero to thirty-one (decimal). Each type of entry has its own form and function. The entry may introduce a major section, provide descriptive information for a section, or give the ID ranges for a section or block. The form of the entries is detailed below. Note that there is no length field for entries which contain ID data. Since the ID data is always the last field in the entry, and since ID bytes always have the sign bit set, the end of the entry can be found by reading the ID bytes until a byte is encountered with the sign bit clear.

Major Subdivisions

0 • End of file.
1 • New author. Followed by a 2-byte length which is the length of the author section (including all nested works). The count includes the length field itself. The length is followed by the 2-byte block number. The block number is the 8K block in which the author begins. The block number is followed by the author ID.
2 • New work. Followed by a 2-byte length which is the length of the work section (including all nested subsections). The count includes the length field itself. The length is followed by the 2-byte block number. The block number is the 8K block in which the work begins. The block number is followed by the work ID.
3 • New section. This marks the next section within the work. Followed by a 2-byte block number. The block number is the 8K block in which the section begins.
4-6 • Undefined.
7 • New file (obsolete). This marks the start of a new file in the combined ID table (also obsolete). Followed by a 2-byte length which is the length of the ID material for this file. The count includes the length field itself. The length is followed by the 4-byte absolute address and the 2-byte length of the text file (expressed in 8K blocks).

ID Fields

8 • Beginning ID for new section. This is the first entry following the new subsection marker (type 3).
9 • Ending ID for new section. This is the last ID entry for the subsection (unless followed by an exception).
10 • Last valid ID for the current block. One of these occurs for each block.
11 • Start exception. This introduces an out-of-sequence ID (i.e. one which does not belong in the current block). The 2-byte block number precedes the ID.
12 • End exception. This gives the end range for the ID exception whose starting range and block number is given by type 11.

13 ◆ Single exception: A single out-of-sequence id.
14 ◆ Undefined.

Descriptive Information

16 ◆ Description of ID fields **a..b**. Followed by a 1-byte identifier (**a..b**=0..1) and a 1-byte length. The length pertains to the description only and does not include the type type, type identifier, or length byte. The description is usually the author or work name. Given at the author or work level, as appropriate. These fields typically indicate the full name of the author and of the works by that author; they should not be confused with the abbreviated forms in the **d** and **c** fields in the actual citations in the texts.

17 ◆ Description of ID fields **n,v..z**. Followed by a 1-byte field identifier. For documents, the n-level identifier = 0, and no other levels are described. For v..z levels, the identifier is 4..0. The identifier is followed by a 1-byte length. The length pertains to the description only and does not include the type type, type identifier, or length byte. Given at the work level. These indicate, e.g., that the **y** level refers to a book of the Aeneid, and the **z** level to a line within that book.

Text in the descriptive fields is assumed to be coded according to the conventions described the document "Beta Coding Summary."

Miscellaneous

18-30 ◆ Undefined

31 ◆ Introduces header of combined ID table. Followed by 3 length bytes, which give the total length in bytes of the combined table. The count includes both the type code byte and the length bytes.

Abbreviated ID Fields

In ID files, the full ID is given for each author ID (level **a** only), each work ID (level **b** only) and each new section (levels **v-z**). In text files the full ID (all levels) is given at the beginning of each 8K block. Other ID fields usually contain only enough information to show how the current ID field differs from the last. Thus most lines in a text require only the code for "increment the z-level" (binary 1000 0000). When the higher levels do not change, they need not be cited. When a higher level does change, all lower levels are implicitly set to binary 1. This often obviates the need to cite the lower levels explicitly. Thus, to cite line 1 of Chapter 2 in a work, the required citation would be "increment the y-level" (binary 1001 0000) since the y-level was previously set for Chapter 1 and the z-level is set to 1

implicitly. Note that when the author, work or or document changes, all lower levels are set to null.

Descriptions in the ID table are handled in a similar fashion. The full description is always given for the author level, the work level, and for the lower levels of the first work. Lower level descriptions for subsequent works are given only as needed. Thus, if thirty consecutive works are cited by Book/line, the table will give this information for the first instance only.

ID Table Sample

In the following sample, binary values are represented as hexadecimal digits; literal ASCII values are given in quotes.

```
07              Type code 7 marks new file
04 f1           Length in bytes of the ID data for this file
00 00 22 08     Absolute address of the text file (in 2K blocks)
00 58           Length of the text file (in 2K blocks)
01              Type code 1 marks new author
02 ac           Length of the author section in bytes
00 00           Author is located in text block 0
ef              Left nibble = 1110: escape code
Right nibble = 1111: ASCII string (no binary)
80              Escape level = 0 (level a)
"0005"          The ID value for level a is ASCII "0005"
ff              ASCII string terminator
10              Type code 16 is author/work description
00              Level = 0 (level a)
0a              Length of description is 10 bytes
"Theocritus"    The author description
02              Type code 2 marks new work
01 d5           Length of the work section in bytes
00 00           Work is located in text block 0
ef              Left nibble = 1110: escape code
Right nibble = 1111: ASCII string (no binary)
81              Escape level = 1 (level b)
"001"           The ID value for level b is ASCII "001"
ff              ASCII string terminator
10              Type code 16 is author/work description
01              Level = 1 (level b)
07              Length of description is 7 bytes
"Idyllia"       The work description
11              Type code 17 is citation description
00              Level = 0 (level z)
04              Length of description is 4 bytes
"line"          The z-level description
11              Type code 17 is citation description
01              Level = 1 (level y)
05              Length of description is 5 bytes
"Idyll"         The y-level description
03              Type code 3 marks new section
00 00           The section is located in text block 0
08              Type code 8 marks the section starting ID
91              Left nibble = 1001: y-level
Right nibble = 0001: literal binary 1
The starting citation is 0005.001.1.1
0a              Type code 10 marks the last ID for the block
8b              Left nibble = 1000: z-level
Right nibble = 1011: 14-bit binary value
81 87           14-bit value is 0000001 0000111 = 135
09              Type code 9 marks the section ending ID
8b              Left nibble = 1000: z-level
Right nibble = 1011: 14-bit binary value
81 98           14-bit binary value is 0000001 0011000 = 152
The ending citation is 0005.001.1.152
```

```
03              Type code 3 marks new section
00 01           The section is located in text block 1
08              Type code 8 marks the section starting ID
92              Left nibble - 1001: y-level
Right nibble = 0010: literal binary 2
The starting citation is 0005.001.2.1
0a              Type code 10 marks the last ID for the block
88              Left nibble = 1000: z-level
Right nibble = 1000: 7-bit binary value
f7              7-bit value is 1110111 = 119
The last ID in block 1 is 0005.001.2.119
09              Type code 9 marks the section ending ID
8b              Left nibble = 1000: z-level
Right nibble = 1011: 14-bit binary value
81 a6           14-bit value is 0000001 0100110 = 166
The ending citation is 0005.001.2.166
...
```

The Author Table (filename AUTHTAB.DIR)

Each entry begins with a file name (without any file name extension) on an even byte boundary. The name is padded with blanks if necessary to reach the fixed length of exactly 8 bytes.

The full author name (of any reasonable length) starts after the file name and is terminated by the first byte value above hex 7f. Up to five synonyms are allowed for each author name.

The (optional) synonym fields are introduced by a byte of hex 80 and are terminated by the first byte value above hex 7f.

The (optional) remarks field is introduced by a byte of hex 81 and is terminated by the first byte value above hex 7f.

The (optional) file size field is introduced by a byte of hex 82 and is terminated by the first byte value above hex 7f.

The (optional) language code field is introduced by a byte of hex 83 and is terminated by the first byte value above hex 7f.

The entry is terminated by at least one hex ff (decimal 255). A second ff is used when needed to pad the entry to an even byte boundary.

If the file name starts with an asterisk, it is a library name (four characters including the asterisk). In this case the second 4 bytes are the binary length of the library (including the 8 bytes for the asterisk, name, and length).

If the file name starts *END, it marks the end of the list. The second 4 bytes are binary zeros.

The text in the AUTHTAB.DIR files (in distinction to the field delimiting informaiton) follows the conventions described in the document "Beta Coding Summary."