TLG CD Format Description

This document is based on the PHI CD ROM Format Description (1992-04-19); the format described herein is intended to be compatible with that of the PHI CD ROMs (including TLG CD ROM #D), but is not to be taken as a duplicate thereof. The format described here applies to TLG CD ROM #E. The format of the CD-ROM has not changed in essence between the two CDs; this document, however, contains clarifications or illustrations of some issues where it has been felt necessary. (See in particular *Sections* and *Exceptions* below.)

CD ROM Format

The CD ROMs issued by the TLG follow the ISO 9600:1988 (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. 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 and the corresponding file name. The entries are arranged alphabetically by filename (i.e. by author number.)

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; on the current TLG CD ROM, the only such files are the canon listings, in bibliographic and database formats (`DOCCAN1.TXT`, `DOCCAN2.TXT`.) For consistency, these texts also carry a citation system (usually a simple line increment; in the case of the canon files, the higher level citations indicate author and work).

Text files are organized in blocks of 8192 bytes. Each block begins with the full citation for the first text line of the block. Subsequent text lines are preceded by an abbreviated citation, as described below. Each text line is terminated with a space (ASCII 20_{hex}), unless the line ends in a hyphen. 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 signaled by an end of block marker in an ID field following the last text line of the block. End of file is indicated by a marker preceding the end of block marker for the final block. Text

lines do not span blocks: if a text line (preceded by its abbreviated citation) will not fit in the current text block, the remainder of the block is padded with null bytes.

Processing a block of text is therefore simple: Read in all bytes with the sign bit set. This is the ID for the first text line. 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 text line. Call a subroutine to process the text. Repeat this process for all text lines in the block, that is, until the ID data contains the end of block marker.

For a description of the text encoding conventions for Greek texts, see the document *Beta Coding Summary*.

An example of a text block follows:

Text: `0001.001, 1.152-155.`

ἈΑρήνηθεν ἔβαν, μεγάλῃ περιθαρσέες ἀλκῇ
ἀμφότεροι· Λυγκεὺς δὲ καὶ ὀξυτάτοις ἐκέκαστο
ὄμμασιν, εἰ ἐτεόν γε πέλει κλέος ἀνέρα κεῖνον
ῥηιδίως καὶ νέρθεν ὑπὸ χθονὸς αὐγάζεσθαι.

```
*)ARH/NHQEN E)/BAN, MEGA/LH| PERIQARSE/ES A)LKH=|
A)MFO/TEROI: *LUGKEU\S DE\ KAI\ O)CUTA/TOIS E)KE/KASTO
O)/MMASIN, EI) E)TEO/N GE PE/LEI KLE/OS A)NE/RA KEI=NON
R(HIDI/WS KAI\ NE/RQEN U(PO\ XQONO\S AU)GA/ZESQAI.
```

Text encoding: `Tlg0001.txt`, block `0001`:

```
00002000: EF80B0B0 B0B1FFEF 81B0B0B1 FFEF82C1 §§§§§§§§§§§§§§§§
00002010: F2E7FF91 8B81982A 29415248 2F4E4851 §§§§§§§*)ARH/NHQ
00002020: 454E2045 292F4241 4E2C204D 4547412F EN E)/BAN, MEGA/
00002030: 4C487C20 50455249 51415253 452F4553 LH| PERIQARSE/ES
00002040: 2041294C 4B483D7C 20804129 4D464F2F  A)LKH=| §A)MFO/
00002050: 5445524F 493A202A 4C55474B 45555C53 TEROI: *LUGKEU\S
00002060: 2044455C 204B4149 5C204F29 43555441  DE\ KAI\ O)CUTA
00002070: 2F544F49 53204529 4B452F4B 4153544F /TOIS E)KE/KASTO
00002080: 20804F29 2F4D4D41 53494E2C 20454929  §O)/MMASIN, EI)
00002090: 20452954 454F2F4E 20474520 50452F4C  E)TEO/N GE PE/L
000020A0: 4549204B 4C452F4F 53204129 4E452F52 EI KLE/OS A)NE/R
000020B0: 41204B45 493D4E4F 4E208052 28484944 A KEI=NON §R(HID
000020C0: 492F5753 204B4149 5C204E45 2F525145 I/WS KAI\ NE/RQE
000020D0: 4E205528 504F5C20 58514F4E 4F5C5320 N U(PO\ XQONO\S
000020E0: 41552947 412F5A45 53514149 2E       AU)GA/ZESQAI.
```

The initial run of ID data (identified by the fact that their sign bits are set: they are all greater than 80_{hex}) gives the full citation for the first text line of this block: `~a0001 b001 c"Arg" y1 z152`. (See below for explanation of the citation formulation.) Then follows line `~z152`, terminated by a space. Then, the abbreviated citation for the next line, `80` (meaning `~z`: increment `z` by one). Then the next line follows, again terminated by a space.

Towards the end of the block, the following obtains:

Text: `0001.001, 1.299-301.`

τῶν μοῖραν κατὰ θυμὸν ἀνιάζουσά περ ἔμπης
τλῆθι φέρειν. θάρσει δὲ συνημοσύνῃσιν Ἀθήνης,
ἠδὲ θεοπροπίῃσιν, ἐπεὶ μάλα δεξιὰ Φοῖβος

```
TW=N MOI=RAN KATA\ QUMO\N A)NIA/ZOUSA/ PER E)/MPHS
TLH=QI FE/REIN. QA/RSEI DE\ SUNHMOSU/NH|SIN *)AQH/NHS,
H)DE\ QEOPROPI/H|SIN, E)PEI\ MA/LA DECIA\ *FOI=BOS
```

Text encoding: `Tlg0001.txt`, block `0001-0002`:

```
00003F90: 494E3A20 8054573D 4E204D4F 493D5241 IN: §TW=N MOI=RA
00003FA0: 4E204B41 54415C20 51554D4F 5C4E2041 N KATA\ QUMO\N A
00003FB0: 294E4941 2F5A4F55 53412F20 50455220 )NIA/ZOUSA/ PER
00003FC0: 45292F4D 50485320 FE000000 00000000 E)/MPHS §.......
00003FD0: 00000000 00000000 00000000 00000000 ................
00003FE0: 00000000 00000000 00000000 00000000 ................
00003FF0: 00000000 00000000 00000000 00000000 ................
00004000: EF80B0B0 B0B1FFEF 81B0B0B1 FFEF82C1 §§§§§§§§§§§§§§§§
00004010: F2E7FF91 8B82AC54 4C483D51 49204645 §§§§§§§§TLH=QI FE
00004020: 2F524549 4E2E2051 412F5253 45492044 /REIN. QA/RSEI D
00004030: 455C2053 554E484D 4F53552F 4E487C53 E\ SUNHMOSU/NH|S
00004040: 494E202A 29415148 2F4E4853 2C208048 IN *)AQH/NHS, §H
00004050: 2944455C 2051454F 50524F50 492F487C )DE\ QEOPROPI/H|
00004060: 53494E2C 20452950 45495C20 4D412F4C SIN, E)PEI\ MA/L
00004070: 41204445 4349415C 202A464F 493D424F A DECIA\ *FOI=BO
00004080: 5320                                 S
```

Line `~y1z299` is normally entered into block `0001`. Since line `~y1z300` with its preceding abbreviated citation (`80`) will not fit into the remainder of block `0001`, the end of block marker (`fe`) is output instead, followed by empty bytes for the rest of the block. At the start of block 0002, the full citation for the new line is given: `~a0001 b001 c"Arg" y1 z300`. The line then follows as normal.

Hyphenated lines appear as follows:

Text: `0003.001,1.1.1.1-2`.

Θουκυδίδης Ἀθηναῖος ξυνέγραψε τὸν πόλεμον τῶν Πελο-
ποννησίων καὶ Ἀθηναίων, ὡς ἐπολέμησαν πρὸς ἀλλήλους,

```
@*QOUKUDI/DHS *)AQHNAI=OS CUNE/GRAYE TO\N PO/LEMON TW=N *PELO-
PONNHSI/WN KAI\ *)AQHNAI/WN, W(S E)POLE/MHSAN PRO\S A)LLH/LOUS,
```

Text encoding: `Tlg0003.txt`, block `0001`:

```
00000061: 402A514F 554B5544 492F4448 53202A29 @*QOUKUDI/DHS *)
00000071: 4151484E 41493D4F 53204355 4E452F47 AQHNAI=OS CUNE/G
00000081: 52415945 20544F5C 4E20504F 2F4C454D RAYE TO\N PO/LEM
00000091: 4F4E2054 573D4E20 2A50454C 4F2D8050 ON TW=N *PELO-§P
000000A1: 4F4E4E48 53492F57 4E204B41 495C202A ONNHSI/WN KAI\ *
000000B1: 29415148 4E41492F 574E2C20 57285320 )AQHNAI/WN, W(S
000000C1: 4529504F 4C452F4D 4853414E 2050524F E)POLE/MHSAN PRO
000000D1: 5C532041 294C4C48 2F4C4F55 532C2080 \S A)LLH/LOUS, §
```

Note that the ID byte (`80`) follows right after the hyphen, without an intervening space. On the other hand, the ensuing line has no final hyphen, and is thus terminated by a space.

Conventionally, any text line with a title citation value (i.e. beginning with `t` or `tit`, optionally followed by at least one digit, or ending with `t` or `tit`, and optionally preceded by comma or at least one digit) is not allowed to be the final citation in a block, and is instead held over to the next block; this should not affect the operation of any browsing software, however. Note that the algorithm for determining what lines

should be held over to the next block, even if they would fit in the current block, is considerably different between TLG CD ROM #D and #E; #D holds lines over much more frequently.

The ID data encoding conventions are detailed below.

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 PHI ID data includes both strictly citation information and unstructured descriptive material; the TLG ID data includes only citation information, but since the descriptive material is optional on PHI CDs anyway, software compatibility between the two is not compromised. Included in the data are codes specifying the ID level (`a... c, v... 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, and in the TLG corpus are always a numerical string, consisting of a four-digit and three-digit number respectively. The `c` level is an optional level specifying the preferred abbreviation for a work. PHI's `d` level, giving the author abbreviation, is not used by the TLG. For example, Apollonius Rhodius' *Argonautica* has the `a-c` level citations of `a = "0001", b = "001",` and `c = "Arg"`. The optional `c` level is not included in the ID Table files, since each work is fully identified by the `a` and `b` levels.

The lower levels, `v` through `z`, are used to cite fields within an individual work. For a given work these behave according to one of two schemes on PHI CDs. The first scheme uses an `~n` level to allow non-hierarchical citations; this scheme is not used by the TLG. In the second, levels `v` through `z` are used strictly 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.

In the following, citations are presented as they are in Beta code, with a preceding tilde (~), and with non-numerical fields enclosed in quotes. Note that `a` and `b` level citations, which need to include leading zeroes, are treated as strings rather than numbers.

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 an ASCII value of `"a12"`.

The ASCII component presently can have a total length from 1 to 15 characters (citations) as used on the TLG CD ROM.

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); `"38"` 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. No such citation currently exists in the TLG databank.

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.

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	8	z-level ID
1001	9	y-level ID
1010	A	x-level ID
1011	B	w-level ID
1100	C	v-level ID
1101	D	n-level ID *(not used on TLG CD ROMs)*
1110	E	Escape code: ID level will be found in next ID byte
1111	F	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	*(not used on TLG CD ROMs)*
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 describe high-level citation ID fields (i.e. author, work). The level code contained in the next byte has for citation IDs the possible values: a = 0, b = 1, c = 2.

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

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. The end-of-ASCII-string code terminates ASCII strings embedded within ID values.

PHI CDs also include exception-start (1111 1000) and exception-end (1111 1001) codes to delimit text lines that appear out-of-order. These codes are not used in TLG CDs; exceptions to the sorting order of citations there are denoted only in the ID table.

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 citation levels, including c if present) is given at the beginning of each 8K block. Other ID fields 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 = hex 80). 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 mark line 1 of Chapter 2 in a (hierarchical) work, the required citation would be "increment the y level" (binary 1001 0000 = hex 90) 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.

Contrary to the PHI documentation, level descriptions (e.g., x = "book", y = "chapter") in the ID table are not handled in a similar fashion, but are always given in full for each new work, whether or not they are the same as for the preceding work.

Coding

On TLG CD ROMs, ASCII information in the citations is treated literally, and the Beta code conventions used in text data are not applicable. In particular, citations may not contain beta escapes: the only brackets admissible are the ASCII brackets, (), [], and {}—not, say, [1]1. Note further that no citation may begin with a non-alphanumeric character. Level descriptions in the ID table, on the other hand, are treated as Beta code strings in Roman font; for example, Galen is cited according to Ku+hn volume/page/line (i.e. *Kühn*). Non-text files on the CD ROM which are by default in Roman font

(i.e. Canon files, canon indices) also use Beta code, but with ASCII parentheses `()` instead of the Beta parentheses `[1 ]1`.

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.

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 need not 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.

Sections

Sections in the ID tables may be defined as follows:

- Any section to the left of `//` in the canon's citation system, as stored in the TLG Canon database. (E.g. if a work has been given a citation system of `Book/section//subsection/chapter/line` in the canon, then any change in `~v` or `~w` is accorded its own section in the ID table.) We will term such a section a major section. While the contents of the citation system appear in the ID table as the level descriptors, the major section distinction is not explicitly noted in any CD file; it is reflected only in the choice of sections.
- If there are no `//` in the canon citation system, any drop in the topmost citation level of the work is accorded its own section. (If the topmost level of the work is unchanged throughout the work, the next highest citation system is considered.)

Therefore a drop in the second level of citation of a work—say from `~w3x3` to `~w3x2`—is *never* accorded its own section in the ID table. This has obvious consequences for traversing the ID table searching for such a citation.

In addition, if a top citation level drop occurs within the range of its surrounding block boundaries, and involves a non-majorsectioned work, it is also not recorded in the ID table. More specifically, if there is a drop $S_1 > S_2$ between block boundaries B_1 and B_2 such that $B_1 < S_2$, $S_2 > B_2$, that drop is not recorded as a section transition. E.g.

B_1 = `~w1x1y3`
S_1 = `~w3`
S_2 = `~w2`
B_2 = `~w4x3y2`

Though a drop from ~w3-~w2 would normally be recorded in the ID table (drop in topmost citation level), it is not recorded in this instance because (a) this is not a work with major sections, so there is no requirement that all transitions in ~w be recorded; and (b) both ~w2 and ~w3 are between the two block boundaries, and can be found as belonging to block B_1-B_2 without further specification.

What all this means is that there is no expectation that the citations within an ID table section be sorted. See for example 0060.003: the following sequence of citations arises:

```
~w29x21
~w29x28
~w29x29
~w29x22
~w29x23
```

The drop from ~x29 to ~x22 is not recorded in the ID table; so if a search is conducted for ~w29x22 in the current block, that search needs to continue past ~w29x28, even though ~x28 > ~x22.

Exceptions

If the citation sought is not found amongst the block boundaries and sections in the ID table (the first half of the ID Table information for any work), it then needs to be sought amongst the exceptions part of the ID table. This proves necessary if there is a drop in citations around the block boundary.

e.g. 0543.001

```
BLOCK 0095:   ~w8x2
              ~w8x"3a"
              ~w8x3y1
              ~w8x3y2
              ~w8x3y3
              ~w8x3y4
              ~w8x3y5
BLOCK 0096:   ~w8x3y6
```

CD ROM #D ends block 0095 on ~w8x"3a", as the largest citation within the block. But on the next block, ~w8x3y6 < ~w8x"3a". The search software would thus expect to find ~w8x3y6 in block 0095; when it does not, it needs to consult the exceptions list for the work, to find that ~w8x3y6 is in block 0096, not 0095.

Note that the listing of exceptions is an unordered list; it should not be expected to be sorted by citation, or even by order of occurrence. Though the listing contains mostly title citations, this is not universally the case, and there are a number of works whose citations are consistently out of order, and where exceptions are essential for smooth browsing through the ID table.

Though the algorithm for determining which citations constitute exceptions has changed between TLG CD ROM #D and #E, this does not affect the way in which exceptions are to be used.

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. (This means that the maximum length for an ID Table is 64K; John Chrysostom comes close to exceeding this limit at over 62K, and this constraint may have to be relaxed in future releases.) The length is followed by the 2-byte block number. The block number is the 8K block in which the author begins; since block numbering starts from zero for each text file, and each author is accorded a new text file, this block number is always 0000. The block number is followed by the author ID string.

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, denoting the 8K block in which the work begins. The block number is followed by the work ID string.

3 New section. This marks the next section within the work. Followed by a 2-byte block number, denoting the 8K block in which the section begins.

4-6 Undefined

7 New file (obsolete). This code is not used on TLG CD ROMs.

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; exception IDs are grouped together at the end of the work information, rather than at the end of each subsection.

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 block one would predict by traversing the intermingled list of block-final IDs (type 10) and subsections (type 8, 9) for the work). 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 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 c fields in the actual citations in the texts,

17 Description of ID fields v... z. Followed by a 1-byte field identifier. 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 identifier, or length byte. Given at the work level. These indicate, e.g., that the y level refers to a book of the *Iliad,* and the z level to a line within that book.

Note that for John Chrysostom (author 2062), length constraints on the ID Table have forced field descriptions to be abbreviated.

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 Header of combined ID table. Not used on TLG CD ROMs.

ID Table Sample

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

`Tlg0005.idt:`

```
00000000: 0101F700 00EF80B0 B0B0B5FF 10001426     ..          . &
00000010: 31546865 6F637269 74757326 20427563 1Theocritus& Buc
00000020: 6F6C2E02 008E0000 EF81B0B0 B1FF1001 ol. . ..
00000030: 07496479 6C6C6961 11010549 64796C6C  Idyllia   Idyll
00000040: 1100046C 696E6503 00000891 0A8B8187  . line ..
00000050: 0A9088F7 0A9488A1 0A9088E0 0A9788A1
00000060: 0A908890 0A988A88 900A988C 888F0A98
00000070: 8E88A90A 9088E10A 9088DB0A 908B8183
00000080: 0A989588 950A9088 D70A8B81 DF0A9898
00000090: 88CE0A90 88C60A8B 81DA0A98 9B88A40A
000000A0: 989F8709 8AA1A8BF A9FF0D00 00918FF4              ..
000000B0: FF02003C 0014EF81 B0B0B2FF 10010B45   .<.           E
000000C0: 70696772 616D6D61 74611101 07457069 pigrammata   Epi
000000D0: 6772616D 1100046C 696E6503 00140891 gram . line .
000000E0: 0A999BAA 8209840D 00149FF4 FF                    .
```

`01` Type code 1 marks new author
`01f7` Length of the (author-specific) ID table (`Tlg0005.idt`) in bytes
`0000` Author starts at block 0 of `Tlg0005.txt`
`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`)
`14` Length of description is 20 bytes
`"&1Theocritus& Bucol."`
The author description
`02` Type code 2 marks new work
`008e` Length of the work section in bytes
`0000` 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
01          Level = 1 (level y)
05          Length of description is 5 bytes
"Idyll"     The y-level description
00          Level = 0 (level z)
05          Length of description is 5 bytes
"line"      The z-level description
03          Type code 3 marks new section
0000        The section is located in text block 0
08          Type code 8 marks the section starting ID
9           y-level
 1          literal binary 1
```

Starting citation: ~y1z1

```
0a          Type code 10 marks the last ID for the block
8           z-level
 b          14-bit binary value
8187        14-bit value is 000 0001 000 0111 = 135
```

End of block 0000: ~y1z135

```
0a          Type code 10 marks the last ID for the block
9           y-level
 0          increment by one
8           z-level
 8          7-bit binary value
f7          7-bit value is 111 0111 = 63
```

End of block 0001: ~y2z63

```
[...]
09          Type code 9 marks the section ending ID
8           z-level
 a          7-bit binary value + ASCII string
a1          7-bit value is 010 0001 = 33
"(?)"       The ASCII string
ff          ASCII string terminator
```

End of section: ~[y31]z"33(?)"

```
0d          Type 13 marks single line exception
0000        This exception is located in block 0
9           y-level
 1          literal binary 1
8           z-level
 f          no binary + ASCII string
"t"         The ASCII string
ff          ASCII string terminator
```

Exception: ~y1z"t" (which precedes the section start, ~y1z1)

```
02          Type code 2 marks new work
003c        Length of the work section in bytes
0014        Work is located in text block 20
ef          Left nibble = 1110: escape code. Right nibble = 1111: ASCII string, no binary
81          Escape level = 1 (level b)
"002"       The ID value for level b is ASCII "002"
```

`ff` ASCII string terminator
`10` Type code 16 is author/work description
`01` Level = 1 (level b)
`0b` Length of description is 11 bytes
`"Epigrammata"`
The work description

Format differences

The 1992 PHI CD format document introduced several features applicable to PHI CD ROMs; these features were not applied to TLG CD ROMs, but are outlined here for clarification.

1. TLG CD ROMs do not use the `~n` ID level; citations on TLG CD ROMs are always strictly hierarchical.

2. TLG ID data does not contain information other than the traditional citation (such as dates).

3. The algorithm used to determine which lines in a text file constitute exceptions is at variance with the algorithm applied to TLG CD ROM #D; as result, the same text files need not generate the same distribution of block endings and exceptions.

4. TLG CD ROMs do not use the `~d` ID level to denote the preferred author abbreviation.

On the other hand, the following 1992 changes do apply to TLG CD ROMs:

1. The information in the ID table is not sorted, but is given in the same order it appears in the text file; in order to locate a specific passage, the entire ID table must be considered. As detailed above, the one exception to this is in the ordering of exceptions, which appear in the ID table in the order of the text blocks they appear in, but whose order within those blocks may not be reflected in the ID table.

2. Single-line exceptions are distinguished from multiple-line exceptions.

3. Citations may not begin with a non-alphanumeric character; a citation value with a leading `*`, `[`, `(`, or `<` is converted to a trailing `*`. Thus, `~x"*4"` is converted to `~x"4*"`, and `~x"(5)"` to `~x"5*"`.

4. The changes in Beta code made as of TLG CD ROM #D still apply: the beta escapes `%1`, `%4`, `%20`, `%21`, `%22`, `%23` in Roman font have been converted to `?`, `!`, `/`, `\`, `=`, `+`, and escapes `#1100-#1199` have been converted to `#1500-#1599`.

Directory Files

AUTHTAB.DIR

The Author Table — a directory of all text files on the CD arranged alphabetically by author name. Include file name, author name, author synonyms, remarks, file size, and language code.

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 (decimal 127).

An (optional) synonym for the author name is introduced by a byte of hex 80 and is terminated by the first byte value above hex 7f. Up to five synonyms are allowed for each author name.

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 four 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 four bytes are binary zeroes.