TCW_#_α-0»:¥TRUNK¥MULTI-IO¥TEXT¥UTF-7.rtf
G+2013-05-03-5.zz.h02:33:

Terry Carl Walker
1739 East Palm Lane
Phoenix, AZ 85006-1930
United States
1-480-929-9628
waxymat@aztecfreenet.org

Proposal for UTF-7 Standard

Proposal for UTF-7 Standard

Occasionally, an ASCII (American Standard Code for Information Interchange) or Unicode file needs to be decoded into standard ASCII symbols in order to see what a file really contains or what a program is doing in creating a file. The proposal in this document creates a new seven bit (BInary digiT) per eight bit byte standard (with each byte representing one word) that displays every Unicode code point in terms of one through four ASCII standard symbols. The following terms in this document are needed in order to describe the resulting shorthand.

Choose one: A list of items within a pair of braces ({}) with the individual items separated by commas. The list of items may include a range of items with the extreme values separated by a colon. Two lists of items that are equivalent will each be enclosed in angle brackets (<>) with only a space (or nothing) between the two lists.

Optional: An item or group of items inside a pair of brackets ([]). Everything inside the brackets is optional.

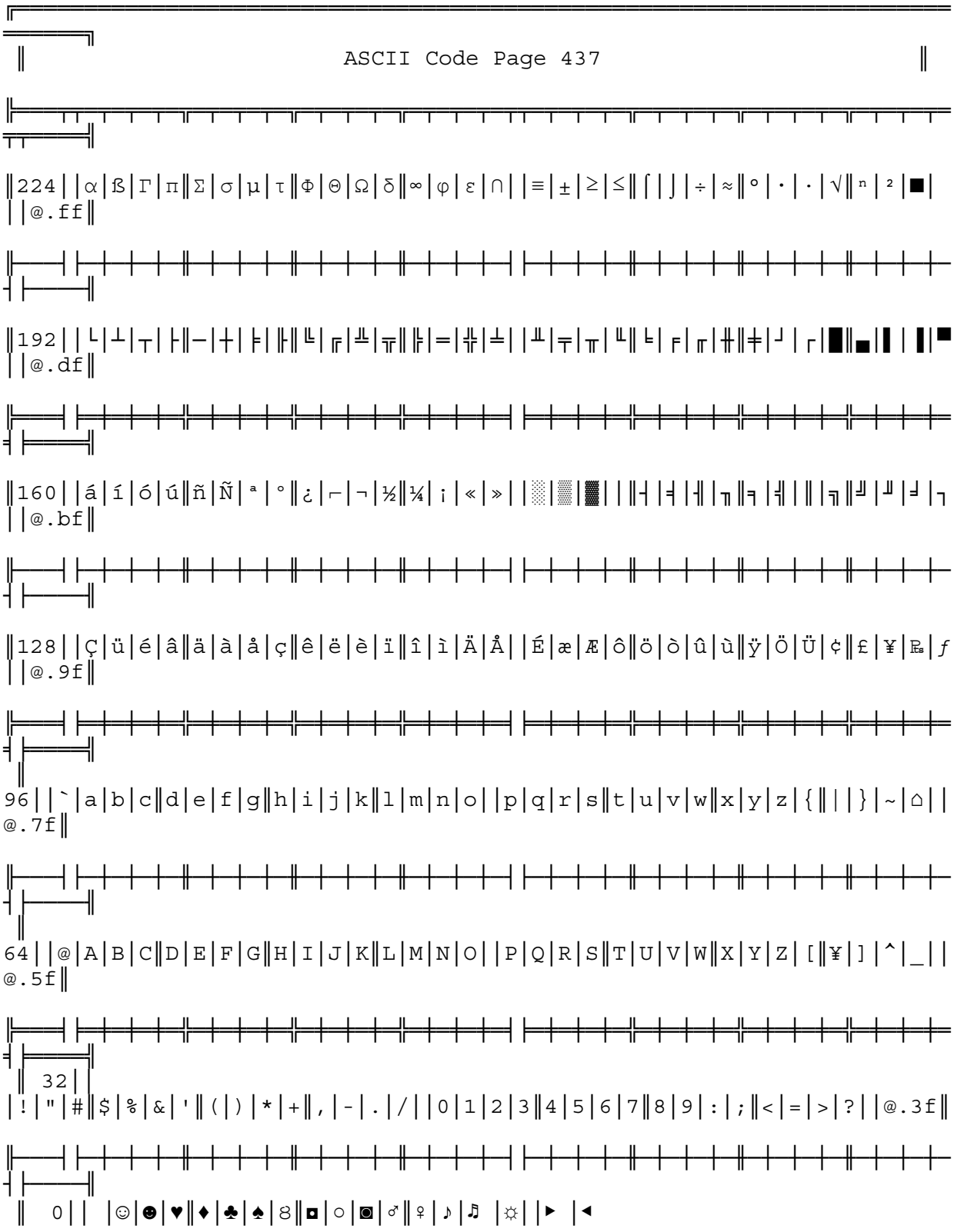
Comment: An item, group of items, or message that is not part of the list or text and can be ignored. It is enclosed inside a pair of paranthesis ("()").

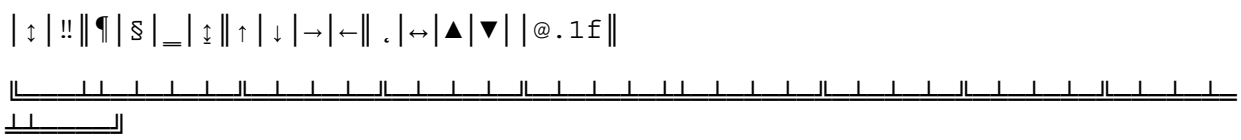
Binary coding: A group of symbols starting with "1@." followed by display characters in which each symbol (except an accent (`)) represents a binary digit position whose value is either zero (0) or one (1). An actual binary digit represents a constant. A letter (<a:z><A:Z>), a question mark (?) or a number sign (#) represents one binary digit of either value. All number signs have the same binary digit value. The letters and question marks have unrelated values within a number even when the same letter or question mark is repeated. However, when the entire string of identical letters show up in two related numbers, the corresponding string of binary digit values is the same in both numbers. An accent is used to separate groups of four binary digits (for easier reading) starting from the extreme right. Two consecutive accents are used to separate groups of sixteen binary digits (again, for easier reading) starting from the extreme right.

Hexidecimal coding: A group of symbols starting with "@." followed by a group of hexidecimal digits and or question marks in which each symbol (digit or question mark) represents four binary digits. The hexidecimal digits are {0:9,<a:f><A:F>}. An actual hexidecimal digit is a constant. A question mark repersents a hexidecimal digit of any value. An accent is used to separate groups of four hexidecimal digits (for easier reading) starting from the extreme right. If the number contains five hexidecimal digits, the highest order (lestmost) hexidecimal digit may instead be an expanded digit with the additional symbols {<g:h><G:H>} that represents higher valued digits.

ASCII: A set of characters whose character values are represented by eight binary digits or two hexidecimal digits. Each character value refers to a character description rather than an actual shape (called the font) or size (usually measured in points). A reference to a character will refer to an ASCII character. All such character values are described with two hexidecimal digits or eight binary digits. All characters have the same height but can vary in width, creating a rectangular cell for each character. The character meanings are standardized for the range 1@.0???`???? (standard ASCII) but vary according to the code page number for the ASCII range 1@.1???`???? (extended ASCII). The standard ASCII characters are further divided into the control characters 1@.000?`???? (which tell devices what to do rather than displaying a character), the space @.20 (" "), the symbol characters @.21:@.7e (which displays a symbol against the background cell in which the symbol does not touch at least two adjacent edges of the cell -- the same edges for all such symbol characters with the possible exception of the underscore @.5f ("_") (depending on the font)), and the delete @.7f (which is not completely defined). Depending on the code page, the delete character and some or all of the control characters may or may not have symbols associated with them -- and the symbols may be different for different code pages. The standard symbol characters are further divided into the Arabic-based decimal digits @.30:@.39 (0:9) which are hereby called the Low digits or Lows, the uppercase Roman letters @.41:@.5a (A:Z), the lowercase Roman letters @.61:@.7a (a:z) and the special symbols which comprise all of the remaining standard characters.

ASCII Code Page 437





Unicode: The Unicode Transform Format (UTF) which assigns character descriptions to various values called code points. They are called code points rather than characters because some characters are formed from a base character followed by zero or more add-on characters. Since Unicode is designed to represent the world's written characters (present and past), it is divided into 17 planes (numbered from zero through sixteen) of sixteen bits each, with the base plane (Basic Multilingual Plane or BMP) as plane zero and the next plane as map zero, the map numbers going up consecutively along with the plane numbers. Map numbers are therefore always one standard hexidecimal digit or four bits long. A code point value is represented as a five digit hexidecimal number (with an extension digit) or as a twenty-one bit number. It should be noted that map numbers above fifteen and the surrogate code points 1@.0`0000``1101`1???`????`???? are not available in UTF-7 and UTF-16 (sixteen bits or two bytes per word). Although other code points may be invalid or unassigned in the standard, all Unicode formats can still represent them. Any unspecified highest order (leftmost) binary or hexidecimal Unicode digits are assumed to have a digit value of zero. Since the lowest code point value descriptions correspond exactly with the standard ASCII value character descriptions, Unicode can be thought of as a (very) expanded version of standard ASCII.

In this document, the binary version of a code point has a generalized form of 1@.t`tttt``uuuu`vwww`wxyz`zzzz. The map number 1@.rrss corresponds to a plane number of 1@.t`tttt minus one.

The ASCII character codes are divided into several catagories for the purpose of defining a UTF-7 standard. Any one ASCII character code belongs to one and only one catagory. The final status of a character code depends on both its catagory and in how it is used. All control codes (1@.000?`????), the delete code (@.7f), and all extended codes (1@.1???`????) are in the Dummy catagory which always gets the status Ignore (as the character is always treated as if it was not there) because the character does not have a standard symbol (or has no symbol), would be displayed differently (or not at all) under different page codes, and may therefore not be visible or may be misinterpreted -- thereby violating the reason for the existance of the UTF-7 standard. The space (@.20) (" ") is also a Dummy character because, without a symbol in its cell, it is sometimes difficult to tell how many spaces exist in a string of spaces. Also, it is often impossible to visually detect spaces at the end of a line. All other character codes are standard symbol display codes. The catagory of High digits (or Highs), when the final status is Hexagon, uses the least significant (rightmost) six bits of a character code as a bit string that is concatenated with other bit strings to create the Unicode code point. The range is 1@.01??`???? (except delete (1@.0111`1111)) (this includes all of the uppercase (@.41:@.5a) (A:Z) and lowercase (@.61:@.7a) (a:z) Roman letters) but also includes 1@.0011`1111 ("?") since the delete character is not available. There is only one character in the Stop catagory -- the colon (@.3a) (":"). Characters in the Start catagory, when the character has the proper number of High characters following it, starts an escape sequence that translates a group of characters into a single code point. In this case, the final status of the Start character is the Begin character. If the Start character and any High digits after it cannot form an escape sequence because the sequence ends prematurely with a Stop character, the Stop character becomes an Ignore character. Excluding the Stop character in this situation, any standard display symbol that is not part of a valid escape sequence is given the final status of a Lone character. An escape sequence always starts with a Start character and always translates into only one code point. Any standard symbol characters that are not in one of the other catagories is in the Core catagory and always has the final status of a Lone character. The value of the Lone character code becomes the value of the corresponding code point. Note that a Stop character can end a potential escape sequence even when the escape sequence would have been invalid anyway. Combining this with the fact that a Stop character can be positioned in different places in a potential escape sequence, there is often two or more ways to encode the same set of Unicode code points. Although it is more ASCII characters, it is recommended (for ease of implementation and for reading clarity) to insert a Stop character after every Start character that is to be interpreted as a Lone character. Also note that when a Stop character is needed as a Lone character when the Stop character would otherwise be interpreted as an Ignore character, two Stop characters are actually needed.

The Start catagory is split into two sub-catagories. Start symbols requiring three High digits to form a valid escape sequence are called Triples. They have the value 1@.0011`11rr (except "?" (1@.0011`1111)) (<,=,>) but also includes 1@.0011`1011 (;) since the "?" character is not available. The complete sequence for a Triple is 1@.0011`1?rr 1@.0?ss`uuuu 1@.0?vw`wwwx 1@.0?yz`zzzz (where 1@.rrss is the map number) creating the code point 1@.t`tttt``uuuu`vwww`wxyz`zzzz. Note that this covers all code points except the BMP. Start symbols requiring two High digits to form a valid escape sequence are called Doubles. They have the value 1@.0010`uuuu (except space " " (1@.0010`0000)) (!,'"',#,$,%,&,"'",(,),*,+,",",-,.,/) but also includes 1@.0011`0000 (zero "0") since the space character is not available. The complete sequence for a Double is 1@.001?`uuuu 1@.0?vw`wwwx 1@.0?yz`zzzz creating the code point 1@.0`0000``uuuu`vwww`wxyz`zzzz. To avoid encoding a surrogate code point, the two character escape sequence 1@.0010`1101 (-) 1@.0?1z`zzzz is defined as valid (when the second character is a High digit) and causes the dash (-) to become a Single Start character, the High digit a Pentagon, and the code point to become the value 1@.0`0000``0000`0000`000z`zzzz -- an ASCII control character. In combination with ASCII standard symbol characters, these escape codes cover the entire BMP except the space and the delete. To avoid a duplication of ASCII standard characters, the two character escape sequence 1@.0011`0000 (0) 1@.0100`000# (@,A) is defined as valid and causes the zero (0) to become a Single Start character, the High digit a Mono, and the code point to become the value 1@.0`0000``0000`0000`0#1#`#### (space, delete). This methodology results in all valid Unicode code points being represented in one through four UTF-7 bytes -- one or two bytes for code points up to seven bits, three bytes for code points from eight bits through sixteen bits, and four bytes for any code point outside of the BMP. This system guarantees that if a UTF-7 file is examined starting from an arbitrary byte position, at most only four standard symbol characters need to be examined in order to determine the corresponding code point value -- up to three standard symbol characters back and or up to three standard symbol characters forward.

In laying out the symbols for the catagories and sub-catagories, an attempt was made to use character values that were as contiguous as possible. Since this meant that the High digits were going to cover at least one band of Roman letters (uppercase or lowercase), the decision was made to cover both bands. Similiarly, the Doubles were either going to cover all of the Low digits or only the digit zero. Avoiding the use of Low digits as Start symbols as much as possible would make it easier to read a UTF-7 file with a lot of numbers (for instance, in a table of numbers). For the same reasons, the Triples and then the Stop symbol were chosen accordingly. Note that all of the Low digits (except zero (0)) -- and only Low digits -- are Core symbols. Also note that all of the Core symbols are contigous.

The one exception to the idea that a Dummy character can appear anywhere in a UTF-7 file is in the Byte Order Mark or BOM. A true BOM (which in UTF-7 is "/{?" (1@.0010`1111 1@.0111`1011 1@.0011`1111) (1@.0`0000``1111`1110`1111`1111) has no preceeding or embedded characters. If the first Unicode code point in the file has this value and has any preceeding or embedded Dummy characters in it, then it is a fake BOM and should be treated as the first actual text code point. Any conversion to another format (including UTF-7) would have to insert a true BOM (no preceeding or embedded Dummy characters) before the first byte of the old file. A true BOM simply tells a program what text format that the file is in and alerts the program that the file is in Unicode. The true BOM is not considered to be an actual text code point. If the first code point of a file does not translate to that of a BOM, then any translation should not insert a BOM at the beginning of the new file.

When two UTF-7 files are concatenated, a true BOM in the second file needs to be removed. Care should also be taken to avoid turning a failed escape sequence at the end of the first file into a valid one as a result of the concatenation by inserting a Stop character between the two files if necessary.

When a UTF-7 file is split, the split should be between two adjacent code points. Care should be taken to avoid turning a Stop character from an Ignore character into a Lone character. A true BOM should be added to the beginning of the split file if and only if the original file has (or should have) a true BOM. Be careful not to turn a fake BOM into a true BOM if the file is split there.

Formatting a UTF-7 file simply means inserting Dummy characters in various places (except a true BOM) into either the file itself or a display of the contents. The most common method is to insert a carriage return (@.0d) (control-M) followed by a line feed (@.0a) (control-J) before the first display symbol that will not fit on the end of the line so that it instead appears as the first character of a new line (wraparound). This creates a block listing of the file without regard to any interpretation of the file contents. A second method involves also repeatedly inserting a space after a fixed number of symbol characters and repeatedly adding a blank line after a fixed number of lines. This is a table listing. Numbering the lines is not an option in UTF-7 since the visible symbols forming the numbers would be mistaken as part of the valid text. Another method involves a variation of the block display by inserting a space (or a new line if at the end of the line) before every character (other than a true BOM) that starts a new code point value -- the interpreted file. Near the end of the page, a form feed (@.0c) (control-L) may be inserted in order to move to the top of the next page or to the top of the screen.

UTF-7 Character Catagories:

```
extended:  1@.1???`????.
delete:  . 1@.0111`1111.
Hexagon: . 1@.01??`???? (not 1@.0111`1111 (delete)) or 1@.0011`1111 (?).
           (@,A:Z,[,¥,],^,_,`,a:z,{,|,},~,?).
Pentagon:  1@.011z`zzzz (not 1@.0111`1111 (delete)) or 1@.0011`1111 (?).
           (`,a:z,{,|,},~,?).
Mono:    . 1@.0100`000#.
           (@,A).
High:  . . Hexagon, Pentagon, or Mono.
Triple:  . 1@.0011`11rr (not 1@.0011`1111 (?)) or 1@.0011`1011 (;).
           (<,=,>,;).
Stop:  . . 1@.0011`1010.
           (:).
Core:  . . 1@.0011`qqqq (range:  1@.0001 ≤ 1@.qqqq ≤ 1@.1001).
           (1:9).
Double:  . 1@.0010`uuuu (not 1@.0010`0000 (space)) or 1@.0011`0000 (zero).
           (zero,!,'"',#,$,%,&,"'",(,),*,+,",",-,.,/)
Single:  . 1@.0010`1101 (-) or 1@.0011`0000 (zero).
           (-,zero).
Start: . . Triple, Double, or Single.
space:     1@.0010`0000.
           (" ").
control: . 1@.000z`zzzz.
Note:  . . 1@.t`tttt = 1@.rrss+1.
```

Unicode Code Points to UTF-7 ASCII Characters:

```
1@.0`0000``1111`1110`1111`1111 =  (Byte Order Mark (BOM))
   1@.0010`1111 1@.0111`1011 1@.0011`1111  (/{?).
      Double      Hexagon     Hexagon.
1@.t`tttt``uuuu`vwww`wxyz`zzzz =  (Range:  1@.0`0001 ≤ 1@.t`tttt ≤
1@.1`0000)
   1@.0011`1?rr 1@.0?ss`uuuu 1@.0?vw`wwwx 1@.0?yz`zzzz (1@rrss =
1@t`tttt-1)
      Triple      Hexagon     Hexagon     Hexagon.
1@.0`0000``uuuu`vwww`wxyz`zzzz =  (Range:  1@.1110 ≤ 1@.uuuu ≤ 1@.1111)
   1@.001?`uuuu 1@.0?vw`wwwx 1@.0?yz`zzzz.
      Double      Hexagon     Hexagon.
1@.0`0000``1101`1www`wxyz`zzzz =  (Surrogate code points -- invalid).
1@.0`0000``uuuu`vwww`wxyz`zzzz =  (Range:  1@.0000`1 ≤ 1@.uuuu`v ≤
1@.1101`0)
   1@.001?`uuuu 1@.0?vw`wwwx 1@.0?yz`zzzz.
      Double      Hexagon     Hexagon.
1@.0`0000``0000`0www`wxyz`zzzz =  (Range:  1@.000`1 ≤ 1@.www`w ≤ 1@.111`1)
   1@.0011`0000 1@.010w`wwwx 1@.0?yz`zzzz  (0  ).
      Double      Hexagon     Hexagon.
1@.0`0000``0000`0000`0111`1111 =  (delete)
   1@.0011`0000 1@.0100`0001  (0A).
      Single       Mono.
1@.0`0000``0000`0000`01yz`zzzz =  (not 1@.yz`zzzz = 1@.11`1111 (delete))
   1@.01yz`zzzz.
      High.
```

Unicode Code Points to UTF-7 ASCII Characters (continued):

```
1@.0`0000``0000`0000`0011`1111 =  (?)
   1@.0011`1111  (?).
      High.
1@.0`0000``0000`0000`0011`11rr =  (not 1@.rr = 1@.11 (?))
   1@.0011`11rr 1@.0011`1010  ( :)
      Triple       Stop.
1@.0`0000``0000`0000`0011`1011 =  (;)
   1@.0011`1011 1@.0011`1010  (;:)
      Triple       Stop.
1@.0`0000``0000`0000`0011`1010 =  (:)
   1@.0011`1010  (:)
      Stop.
1@.0`0000``0000`0000`0011`mmmm =  (Range:  1@.0001 ≤ 1@.mmmm ≤ 1001 (1:9))
   1@.0011`zzzz  (1:9).
      Core.
1@.0`0000``0000`0000`0011`0000 =  (zero)
   1@.0011`0000 1@.0011`1010  (0:)
      Double       Stop.
1@.0`0000``0000`0000`0010`uuuu =  (not 1@.uuuu = 1@.0000 (space))
   1@.0010`uuuu 1@.0011`1010  ( :)
      Double       Stop.
1@.0`0000``0000`0000`0010`0000 =  (space)
   1@.0011`0000 1@.0100`0000  (0@).
      Single       Mono.
1@.0`0000``0000`0000`000z`zzzz =  (control)
   1@.0010`1101 1@.0?1z`zzzz  (- ).
      Single     Pentagon.
```

UTF-7 ASCII Characters to Unicode Code Points:

```
/{? . . .  1@.0010`1111 1@.0111`1011 1@.0011`1111  (BOM (Byte Order Mark))
          ("/{?") =
     1@.0`0000``1111`1110`1111`1111.
extended:  1@.1???`????  (Dummy) (Ignore).
delete: .  1@.0111`1111  (Dummy) (Ignore).
High: . .  1@.01yz`zzzz (not 1@.0111`1111 (delete))
          (@,A:Z,[,¥,],^,_,`,a:z,{,|,},~) =
     1@.0`0000``0000`0000`01yz`zzzz.
High: . .  1@.0011`1111
          (?) =
     1@.0`0000``0000`0000`0011`1111.
Triple: .  1@.0011`11rr (not 1@.0011`1111 (?)) or 1@.0011`1011 (;)
          (<,=,>,;).
Triple Hexagon Hexagon Hexagon =  (1@.t`tttt = 1@.rrss+1)
 1@.0011`1?rr 1@.0?ss`uuuu 1@.0?vw`wwwx 1@.0?yz`zzzz =
     1@.t`tttt``uuuu`vwww`wxyz`zzzz.
Triple Hexagon Hexagon [Stop] =
 1@.0011`1pqq 1@.0ooo`oooo 1@.0nnn`nnnn [1@.0011`1010] =
     1@.0`0000``0000`0000`0011`1pqq
     1@.0`0000``0000`0000`0ooo`oooo
     1@.0`0000``0000`0000`0nnn`nnnn.
Triple Hexagon [Stop] =
 1@.0011`1pqq 1@.0ooo`oooo [1@.0011`1010] =
     1@.0`0000``0000`0000`0011`1pqq
     1@.0`0000``0000`0000`0ooo`oooo.
Triple [Stop] =
 1@.0011`1pqq [1@.0011`1010] =
     1@.0`0000``0000`0000`0011`1pqq.
```

UTF-7 ASCII Characters to Unicode Code Points (continued):

```
Stop: . .  1@.0011`1010.
          (:).
     1@.0`0000``0000`0000`0011`1010.
Core: . .  1@.0011`mmmm  (1@.0001 ≤ 1@.mmmm ≤ 1@.1001) =
          (1:9).
     1@.0`0000``0000`0000`0011`mmmm.
Single Mono =  1@.0011`0000 1@.0100`000# =
          ("0"{@,A})
     1@.0`0000``0000`0000`0#1#`####.
Single Pentagon =  1@.0010`1101 1@.0?1z`zzzz =
          ("-"{`,a:z,{,|,},~,?})
     1@.0`0000``0000`0000`000z`zzzz.
Double: .  1@.0010`uuuu (not 1@.0010`0000 (space)) or 1@.0011`0000 (zero)
          (zero,!,'"',#,$,%,&,"'",(,),*,+,",",-,.,/).
Double Hexagon Hexagon =
 1@.001?`uuuu 1@.0?vw`wwwx 1@.0?yz`zzzz =
     1@.0`0000``uuuu`vwww`wxyz`zzzz.
Double Hexagon [Stop] =
 1@.001l`llll 1@.0ooo`oooo [1@.0011`1010] =
     1@.0`0000``0000`0000`001l`llll
     1@.0`0000``0000`0000`0ooo`oooo.
Double [Stop] =
 1@.001l`llll [1@.0011`1010] =
     1@.0`0000``0000`0000`001l`llll.
space:  .  1@.0010`0000  (Dummy) (Ignore).
control:  1@.000?`????  (Dummy) (Ignore).
```