PBASIC 2.x ENHANCED SYNTAX NOTES:

- New Control Character pre-defined constants
  - To reflect the control characters allowed by the Stamp Windows Editor
  - **LF** (10)
  - **CRSRUP** (5)
  - **CRSRDN** (6)
  - **CRSRLF** (3)
  - **CRSRRT** (4)
  - **CLRDN** (12)
  - **CLREOL** (11)
  - **CRSRXY** (2, must be followed by an X-byte and a Y-byte)

- **IF**..**THEN**..**ELSE**..**ENDIF**
  - Syntax: (Items in brackets '{}' are optional)

    IF condition(s) **THEN** statement(s) { **ELSE** statement(s) }

    —OR—

    IF condition(s) **THEN**
       statement(s)
    { **ELSE** }
       { statements(s) }
    ENDIF
  - Note: multiple statements can be included in the main or else parts of a single-line IF..THEN by inserting colons ":" in between the statements, as in:

    IF condition(s) **THEN** statement1 : statement2 **ELSE** statement3 : statement4
  - Also note: ENDIF only required on multi-line IF..THEN statements.
  - Up to 16 nested IF..THENs allowed.

- **DO**..**LOOP**
  - Syntax: (Items in brackets '{}' are optional)

    DO { {**WHILE** | **UNTIL**} condition(s) }
       statement(s)
    LOOP { {**WHILE** | **UNTIL**} condition(s) }
  - The conditional statement can appear on the DO (to make a 0..N iterative loop) or on the LOOP (to make a 1..N iterative loop) or it can be left off entirely, to make an endless loop.
  - Up to 16 nested DO..LOOPs allowed.

- **EXIT**
  - Causes execution to immediately move to the instruction following the end of the loop.
  - Supported inside **DO**..**LOOP**
  - Supported inside **FOR**..**NEXT**
  - Up to 16 EXITs can appear in any give loop.

- **READ/WRITE** word-sized values
  - Syntax: (Items in brackets '{}' are optional)

    READ location, { **WORD** } variable
    WRITE location, { **WORD** } variable

- **SELECT CASE**
  - Syntax: Note: (|) denotes mutually exclusive items. { } denotes optional items

```
SELECT  expression
   (CASE | TCASE)  ( ELSE | condition(s) )
                                  Statement(s)
   …
ENDSELECT
```

  - expression can be a variable, a constant or an expression.
  - Condition can be of the form:
    - {cond-op} #
      - cond-op is an optional conditional operator: =, <>, <, >, >= or <=
      - # is a variable, a constant or an expression.

    --OR--
    - # TO #
      - Indicates a range of the first number to the next number, inclusive.
      - Conditional operators are not allowed in this form.
  - Multiple conditions within the same case can be separated by commas ",".
  - When a case is true, the default function is for the case's Statement(s) to be executed, then program execution jumps to the first statement following the **ENDSELECT**.
  - **TCASE**, meaning "Through **CASE**", behaves exactly like **CASE**, except that it causes the previous **CASE** (if executed) to continue program execution at the first statement within the **TCASE**, instead of jumping to after the **ENDSELECT**. After execution of the statements within **TCASE**, execution jumps to after **ENDSELECT**, unless followed by another **TCASE**.
- **PIN** type
  - Syntax:
    symbol **PIN** constant-expression
  - Context-sensitive symbol.
  - In situations where you expect to "read" a variable, it acts like INx.
  - In situations where you expect to "write" a variable, it acts like OUTx.
  - In situations where the Stamp expects a constant, it acts like a constant x.
  - Is always a constant when used in "pin" arguments of any command.

- Line-continuation
  - Any line of code can be continued onto the next line by breaking the first line just after the comma "," separating arguments or list items.

    -
      ```
      BRANCH  Idx, [Label1, Label2,
                    Label3, Label4]
      ```

    -
      ```
      DEBUG  "Hello ",
             "World!"
      ```

    -
      ```
      SELECT  X
          CASE  10, 20 TO 40,
                50 TO 60, 100   : HIGH 1     'pin 1 high when X = 10, 20..40, 50..60 or 100
          CASE  > 100           : LOW 1      'Set pin 1 low when X > 100
      ENDSELECT
      ```

- **ON**
  - Syntax: Note: (|) denotes mutually exclusive items. { } denotes optional items
    ON expression (**GOTO | GOSUB**) label {, label…}

- **$PBASIC** directive.
  - Syntax:
    '{$PBASIC #} ;where # is 2.0 or 2.5
  - Version 2.0 is the "classic" tokenizer.
  - Version 2.5 is the "enhanced"tokenizer.

- **#IF**..**#THEN**..**#ELSE**..**#ENDIF** directives
  - Conditional compilation directive. Surround code to include/exclude based on condition.
  - Syntax: Similar to IF..THEN..ELSE..ENDIF. (Items in brackets '{}' are optional)
    #IF condition(s) #**THEN** statement(s) { #**ELSE** statement(s) } **#ENDIF**

    —OR—

    #IF condition(s) #**THEN**
    statement(s)
    { #**ELSE** }
    { statements(s) }
    #ENDIF
  - Condition can contain compile-time constants, defined symbols, numbers, parenthesis and the following operators:
    - =
    - >
    - <
    - <>
    - >=
    - <=
    - AND
    - OR
    - XOR
    - NOT
    - +
    - -
    - *
    - /
    - <<
    - >>
  - Up to 16 nested #IF..#THENs allowed.

.

- **#SELECT #CASE** directives
  - Syntax: Similar to SELECT CASE. Note: (|) denotes mutually exclusive items. { } denotes optional items
    #SELECT expression
    #**CASE** (**#ELSE |** condition(s))
    Statement(s)
    …

#ENDSELECT

- expression can contain compile-time constants, defined symbols, numbers and parenthesis. It can also contain the following operators:
  - +
  - -
  - *
  - /
  - <<
  - >>
- Condition can be of the form:
  - {cond-op} #
    - cond-op is an optional conditional operator: =, <>, <, >, >= or <=
    - # is a variable, a constant or an expression.

  --OR--
  - # TO #
    - Indicates a range of the first number to the next number, inclusive.
    - Conditional operators are not allowed in this form.
- Multiple conditions within the same case can be separated by commas ",".
- When the first case that is true is encountered, the case's Statement(s) are compiled into the code and all other cases are ignored.

- **#DEFINE** directive
  - Defines a pre-compile-time symbol that may be tested using the #IF or #SELECT directives.
  - Syntax:
    #DEFINE symbol { = expression}.
  - expression can contain compile-time constants, defined symbols, numbers and parenthesis. It can also contain the following operators:
    - +
    - -
    - *
    - /
    - <<
    - >>
  - By using the optional expression parameter, a value can be assigned to the defined symbol. For example: #DEFINE Mode = 5 defines a precompiler symbol called Mode that is equal to the number 5.
  - By omitting the optional expression parameter, the symbol is treated as defined. This allows a simple testing method such as:

    ```
    #DEFINE  CompileAll
    …
    #IF  CompileAll  #THEN
    …
    #ENDIF
    ```

    Note that if the first line, the #DEFINE, statement is removed or commented out of the code, the #IF..#THEN statement will evaluate to false (meaning the CompileAll symbol is not defined) and the statements within the #IF..#ENDIF block will NOT be compiled into the code, in this case.

- **#ERROR** directive
  - Creates a user-defined error message.
  - Syntax:
    #ERROR TextString.
  - TextString is a string of characters and ASCII constants that will be displayed as an error message if the #ERROR directive is encountered during compilation.
  - This allows an error for situations that can not be, or are not, handled. For example, if a developer wrote a program that will only work on the BS2e and above, that developer can keep a user from downloading it to a BS2 with the following:

    ```
    #IF  $STAMP = BS2  #THEN
      #ERROR "Sorry, this program will only work on a BS2e or above!"
    #ENDIF
    ```

    If the user ever compiles it for a BS2, the $STAMP precompiler symbol will be set equal to BS2, the #IF..#THEN directive will evaluate to True and the #ERROR directive inside the #IF..#ENDIF block will be executed, generating a compile error with a message, "199-Sorry, this program will only work on a BS2e or above!".

    The "199-" means error message number 199, which is used for user-defined errors.

- Error occurs if **FOR** found without **NEXT**
- Error occurs if **DO** found without **LOOP**
- Error occurs if multiline **IF** found without **ENDIF**
- Error occurs if **LABEL** found without colon
  - Even catches things like PULSEOUT 1, 1000 (PULSOUT is misspelled)… will be thought of as a label and will cause the same error (more clear than in previous tokenizer).
- Disallows overlapping code blocks.