1 Pattern Appendix

1.1 Pattern Variables

variable
signifies a static variable

‘variable‘
signifies an unevaluated variable in a pattern

1.2 Pattern Operators

pattern1 && pattern2 **pattern concatenation**
pattern concatenation operator produces a new pattern containing the left operand followed the right operand.

---

pattern1 .| pattern2 **pattern alteration**
pattern alteration operator produces a pattern containing either the left operand or the right operand.

---

substring -> variable **conditional assignment**
assigns the substring on the left to the variable on the right if the pattern match is successful.

---

result $$ variable **immediate assignment**
assigns the immediate result on the left to a variable on the right within a pattern.

.$ variable **cursor position assignment**
assigns the cursor position of the string to a variable on the right within a pattern.

---

string ?? variable **comparison operator**
compares the string on the left to see if there are any matches of the pattern on the right.

1.3 Pattern Built-In Functions

PAny(s) **match any**
matches any of the subject characters in s.

PArb() **arbitrary pattern**
matches any arbitrary pattern of any length.

PArbno(p) **repetitive arbitrary pattern**
matches repetitive sequences of p in the subject string.

PBal() **balanced parentheses**
matches the shortest non-null string which parentheses are balanced.

PBreak(s) **pattern break**
matches any characters up to but not including any of the subject characters in s.

PBreakx(s) **extended pattern break**
matches any characters up to any of the subject characters in s, and will look beyond the break position for a possible larger match.

PCancel() **pattern cancel**
causes an immediate failure of the entire pattern match.

PFail() **pattern failure**
signals the failure of the current portion of the pattern match.

PFence() **pattern fence**
signals a failure in the current portion of the pattern match if it is trying to backing up to try other alternatives.

PLen(I) **match fixed-length string**
matches a string of a length of I characters.

PNotAny(s) **match not any**
matches any of the subject characters that are not in s.

PPos(I) **cursor position**
sets the cursor position of a string in a pattern measured from the left to the right. the first position precedes the first character in the string and has a value of 1.

PRest() **rest of pattern**
matches the remainder of the subject string.

PRpos(I) **reverse cursor position**
sets the cursor position of a string in a pattern measured from the right to the left. the first position follows the last character in the string and has a value of 0.

PRtab(I) **pattern reverse tab**
matches any characters from the current position up to the specified position to

the right.

PSpan(s) **pattern span**

matches one or more subject characters for the set in s. It must match at least one character.

PSucceed() **pattern succeed**

(may be a frivolous function)
signals a success of the current portion of the pattern match.

PTab(I) **pattern tab**

matches any characters from the current position up to the specified position to the right.