71-00008 PROGRAM DESCRIPTION

Program Title	Customization Utilities (CUSTUTIL)		
Contributor	Hewlett Packard Company		
Address	1000 N.E. Circle Blvd.		
City	Corvallis	State	Or.
Country	U.S.A.		
Telephone	(503)757-2000	Zip/Postal Code	97330

Program Description (include equations) CUSTUTIL provides six keywords that are helpful in customizing the user interface: INLINE gives an enhanced input capability; it allows you to determine the cursor position and type, and which keys terminate MSG$ allows for localization of error messages and user input, making it possible for a Basic program to be translated in to any language automatically. KEYWAIT$ puts the 71 in a low power state, waiting for a key to be hit, then returns key name. SCROLL scrolls the message in the display the specified number of characters. KEYNAM$ returns keyname, given keycode. KEYNUM returns keycode, given keyname.

Necessary Accessories None

Operating limits and warnings

Minimum RAM Requirement 1007 bytes

References

Syntax and Explanation of each Keyword:

INLINE Statement

Syntax:

```
INLINE <input string>,<char# in LCD position 1>,...
 ...<cursor position/type>,<terminators>,<variable1> [,<variable2>...
 ...[,<variable3>] ]
```

<input string>::=	String expression to be displayed as prompt
<char# in LCD pos. 1>::=	Numeric expression which rounds to X, such that: 1<= X <= 96 Value out of range generates error. Determines how many characters of displayed string are scrolled off left end of the display. For example: 1=> no characters scrolled 2=> 1 character scrolled
<cursor pos/type>::=	Numeric expression which rounds to X, such that: 1<= |X| <=96 Value out of range generates error. Determines which character in the display the cursor is on. Regardless of input, this value is forced to be at least as large as the char# of the first readable character in the display; also, it is forced to be no bigger than 1 character position beyond the last readable character in the input string. Negative argument indicates an insert cursor.
<terminators>::=	String expression of the form: #<physical keycode>#<physical keycode>... Keys are numbered in row-major order 1-56. For f-shifted keys, add 56; for g-shifted keys, add 112. Determines which keys terminate INLINE. Null string or string not conforming to syntax above generates error. '#' as last character in string is ignored.
<variable1>::=	Numeric variable into which the terminator number is returned. The variable specified contains 'n' on exit if the terminator hit was the nth specified in the terminator list.
<variable2>::=	Numeric variable into which the final cursor position and type is returned.

Assuming |variable2| = n, the cursor was on the nth character in the 'free portion' of the display buffer. See the discussion of WINDOW in the HP-71 Reference Manual for details.

If <variable2> < 0 , then insert cursor

<variable3>::= Numeric variable into which the character# in LCD position 1 is returned.

Once again, note that WINDOW affects the effective size and location of the LCD.

Description:

INLINE is a statement that extends the capability given in the HP-71's INPUT statement and KEY$ function. INLINE allows you to specify
 a) the prompt string
 b) the number of prompt string characters to be scrolled off the left side of the display
 c) where in the display the cursor is to come up flashing, and
 d) what type of cursor (replace/insert)

INLINE allows the user to press any combination of keys for input and editing, just like the INPUT statement. While INPUT terminates execution only when specific keys are pressed (such as [Endline]), any number of different keys can be defined to terminate INLINE execution. When one of these terminating keys is pressed, INLINE returns a number that indicates which key caused termination; INLINE will optionally return additional values indicating the cursor position/type and number of characters scrolled off the left side of the display on exit.

For increased customization, the input string may contain cursor on and cursor off characters to make certain portions of the string non-editable.

There are three additional limitations placed on the input parameters for <char# in LCD pos. 1> and <cursor pos.>:
 1) If <char# in LCD pos 1> is greater than <cursor pos>, then <char# in LCD pos 1> is set equal to <cursor pos>.
 2) <char# in LCD pos 1> is limited to be <= 97 - WINDOWsize
 3) If <cursor pos> exceeds <char# in LCD pos 1> + WINDOWsize, then the specified <cursor pos> takes precedence, and the <char# in LCD position 1> is incremented until the 'cursor character' appears in the display window.

For example,

```
 INLINE A$,91,80,T$,A
```

According to (1) above, <char# in LCD pos 1> becomes 80, instead of 91. Then, according to (2) above, <char# in LCD pos 1> becomes 75 (assuming the default WINDOWsize of 22).

To illustrate (3) above:

```
 INLINE A$,60,95,T$,A
```

In order to get character #95 in the display window, character #74 (96-22) is put in LCD position 1.

Following is an example illustrating the use of protected fields (non-editable characters) in the <input string>:

```
INLINE CHR$(27)&"<"&"Enter Name "&CHR$(27)&">"&C$,2,1,"#38#50#51",A,B,C
```

Assume that C$ contains the default input string. In this example the user cannot back the cursor up over the prompt since the cursor was turned off. However, they can edit the default input string since the cursor was turned back on. The replace cursor will come up on the first 'readable' character, that is the first character displayed in which the cursor is on (in this example that is the first character of the default input string) -- this was specified by the cursor position/type argument. The first character of the input string will be scrolled off the left side of the display -- this was specified by the next argument.

INLINE will terminate on one of three keys:
[Endline],[Up arrow],[Down arrow]. If [Down arrow] is the terminator key, A=3 on exit. If the user typed in a five character name before hitting the terminator key (assuming no backspaces), B=17 on exit (the cursor originally came up on the 12th character in the display and was advanced 5 more character positions), and C=2.

Note that the <cursor position> argument 'counts' readable characters only. Also, DISP$ 'sees' readable characters only, so that a DISP$ done in the above example returns only the user input (including the default input), not the prompt itself.

Also note that the cursor position argument and the value returned in the first optional variable do not operate totally analogous. The cursor position argument counts readable characters only, whereas the value returned in B (in the example above) reflects the TOTAL number of characters in the "free portion" of the display, readable and non-readable.

Related Keywords:

DISP$, WINDOW

KEYNAM$ Function

Syntax:

```
KEYNAM$(<physical keycode>)
```

```
<physical keycode>::=    Numeric expression, rounded to integer X,
                         such that 1<= X <=168

                         All values out of range (with the exception
                         of zero) generate an error.
                         KEYNAM$(0) returns the null string.
```

Description:

Given the physical keycode (keys are numbered in row-major order), KEYNAM$ returns the corresponding key name. Refer to the KEY$ function in the HP-71 Reference Manual for an explanation of key names.

KEYNAM$ is the complement of KEYNUM.

Examples:

```
KEYNAM$(1)   -- returns Q
KEYNAM$(113) -- returns q
KEYNAM$(57)  -- returns fQ
```

Related Keywords:

KEYNUM

KEYNUM Function

Syntax:

KEYNUM(<key name>)

<key name>::=	String expression Any string that isn't a valid key name generates an error, with one exception: If the string is null, KEYNUM returns 0. Refer to 'key name' in the glossary of the HP-71 Reference Manual for further details.

Description:

Given a key name, KEYNUM returns the corresponding physical keycode. It is the complement of KEYNAM$.

Examples:

```
KEYNUM("Q")     -- returns 1
KEYNUM("fQ")    -- returns 57
KEYNUM("#113") -- returns 113
```

Related Keywords:

KEYNAM$

KEYWAIT$ Function

Syntax:

KEYWAIT$

Description:

When the KEYWAIT$ function is executed, the HP-71 goes into a low power consumption state until a key is pressed; when a key is pressed, KEYWAIT$ returns the corresponding key name.

Related Keywords:

KEY$

MSG$ Function

Syntax:

MSG$(<lllmmm>)

where lll is the three-digit LEX file ID
  and mmm is the three-digit message number.

If the specified LEX file doesn't exist, or if the specified message number does not exist in the LEX file, MSG$ returns the null string.

Description:

MSG$ allows a BASIC user to build custom messages from any message table. In addition, the translation capability provides a powerful tool for BASIC application pacs to accept commands in any language. An excellent example is the HP-71 Text Editor, a BASIC program that stores all its commands, responses, and HELP catalog information in a message table. All user input is compared to entries in the message table, using MSG$.

To build your own foreign language LEX file, refer to MSG$ in the HP-71 IDS Volume I.

Examples:

```
DISP MSG$(255131)  -- displays message number 131 from LEX file 255,
                      according to the foreign language LEX file that
                      is currently plugged in.

DISP MSG$(085001)  -- displays the first message from LEX file 85
```

SCROLL Statement

Syntax:

SCROLL <char# in LCD pos. 1>

<char# in LCD pos. 1>::= Numeric expression, rounded to an integer value.
Errors if negative.

Description:

The SCROLL statement scrolls the message in the display the necessary number of characters, so that the character you specify appears in LCD position 1.

The number of characters can be specified by any positive numeric expression. An error results if the rounded integer value is negative, or if it exceeds 1,048,575 (FFFFF Hex).

For a rounded integer value of 0, SCROLL interprets the parameter as 1.

Related Keywords:

WINDOW