THE TYPESCRIPT MANAGER

December 7, 1984

2600 Liberty Avenue
P. O. Box 2600
Pittsburgh, PA 15230
(412) 355-0900

Accent is a trademark of Carnegie-Mellon University.

Accent and many of its subsystems and support programs were originally developed by the CMU Computer Science Department as part of its Spice Project.

The information in this document is subject to change without notice and should not be construed as a commitment by PERQ Systems Corporation. The company assumes no responsibility for any errors that may appear in this document.

PERQ Systems Corporation will make every effort to keep customers apprised of all documentation changes as quickly as possible. The Reader's Comments card is distributed with this document to request users' critical evaluation to assist us in preparing future documentation.

PERQ, PERQ2, LINQ, and Qnix are trademarks of PERQ Systems Corporation.

Table of Contents Page

1. Theory

The Typescript Manager maintains standard text windows, called typescripts. The Typescript Manager provides a number of facilities, including the handling and editing of input, modes of text display, and cursor addressing.

1.1. Input Facilities

The Typescript Manager allows the user to edit input lines using a subset of the commands available in the system editor. All the editor's single-line editing commands are available. The user also has the ability to recall past input lines to either reissue them or edit them into new input lines. A list of all available editing commands is in Chapter 4.

Escape completion is provided by the Typescript Manager through the File System. The user may type a partial filename and then the Escape key; the Typescript Manager will ask the File System to complete the file name by finding the longest unambiguous match to the partial name and placing it in the input line.

1.2. Display Facilities

The Typescript Manager provides two modes of display, "More" mode and "Continuous Scroll" mode. The user can select which mode is to be used, using the editing functions in Chapter 4. A program can turn more mode on or off, using the STSMoreMode routine (Section 3.2.27).

In "More" mode, when a full window of output has been displayed, a black bar appears at the bottom of the window and the output stops. Pressing the LineFeed key will display the next

windowful of output.

In "Continuous Scroll" mode, output is displayed continually until it has all been displayed. The user can use the process control functions "Suspend Process" and "Resume Process" to stop and start output.

The Typescript Manager also remembers the last several pages of text output by a program using a typescript. When the text window changes shape, size, or coveredness, the Typescript Manager will redisplay the changed portion of the window. It will add text that had previously scrolled off the window if needed. This stored information can also be viewed by using some of the editing functions available.

1.3. Cursor Addressing

The Typescript Manager provides a set of routines that allow a user program to control the location of the cursor in a viewport. This is known as cursor addressing. Cursor addressing allows the cursor to be moved from position to position. There are a number of operations available with cursor addressing.

1.4. Windows and Viewports

Typescripts can be set up in both windows and viewports. The key difference between a window and a viewport is that input from the keyboard is received in a window, while a viewport is used to display text being output from a program. See the document "The Window Manager" in this manual for a complete discussion of the difference between windows and viewports.

2. Use

Each typescript maintained by the Typescript Manager has its own port. All requests for input, output, and control of a typescript are directed to its port. There is a master Typescript Port used to create new typescripts.

When a user program is started, the master Typescript Port is TypescriptPort and the port associated with the program's window is UserTypescript. Both of these ports are defined in PascalInit.Pas in LibPascal. Pascal input and output through the default files INPUT and OUTPUT is directed to UserTypescript (unless they have been redirected).

A program can create a new typescript in an existing window or viewport by calling one of the following routines:

STSOpen
STSOpenWindow
STSFullOpen
STSFullOpenWindow
STSOpenTerm

Each of these routines will return a port for the typescript in that window or viewport.

The Typescript Manager provides routines to read input from a typescript in a window. They allow the user to type either a character or a line of input, using the available line editing commands. Input lines are completed by typing Return.

A user program can output text to a viewport as either a character, a string, or an array. When outputting a string or an

array, a linefeed character will end a line of text, scrolling the typescript and putting the next character at the beginning of the next line.

The STSOpenTerm routine opens a typescript in a viewport with cursor addressing. This allows the user program to do cursor addressing operations in the typescript. There are three types of cursor addressing operations: cursor operations, delete operations, and screen operations. These are described in Sections 3.2.20, 3.2.21, and 3.2.22.

Other Typescript Manager routines allow user programs to take control of windows, change the environment of a typescript, change key translation tables, and perform other operations.

3. Definitions

The definitions throughout this document are given in Pascal. If you are programming in the C language, please refer to the document "C System Interfaces" in the *Accent Languages Manual*. If you are programming in the Lisp language, see the document "Lisp Interaction with the Accent Operating System" in the *Accent Lisp Manual*. When FORTRAN becomes available under Accent, the definitions will be the same as in the C language.

3.1. Type Definitions

The following type definitions are module TSDefs in TSDefs.Pas in LibPascal.

```
const
     { constants designate cursor
     { addressing operations }
  ts_home = 0;
  ts_boln = 1;
  ts_eoln = 2;
  ts_up = 3;
  ts_down = 4;
  ts_left = 5;
  ts_right = 6;
  ts_delchar = 0;
  ts_ereoln = 1;
  ts_erboln = 2;
  ts_bell = 0;
  ts_clear = 1;
  ts_redisplay = 2;
  ts_scroll = 3;

type
  CursorOp = 0 .. 6;
  DeleteOp = 0 .. 2;
  ScreenOp = 0 .. 3;
   { variable-length array used for passing long strings: }
  TSCharArray = packed array [0 .. 0] of char;
  pTSCharArray = ^TSCharArray;
  TString255 = string[255];
  Typescript = port;
```

3.2. Routine Definitions

The routine definitions are found in module TS in TSUser.Pas in LibPascal.

3.2.1. Returning version number

```
Function Typescript_Version(
                ServPort : Port
        ): TSString255;
```

Abstract:

Returns a string giving the version number and date of the Typescript Manager.

Parameters:

ServPort Any port

Returns:

Version number and date in the form "v_ of dd-mmm-yy."

3.2.2. Creating typescript in a viewport with defaults

```
Function STSOpen(
            ServPort : Port;
            vp       : ViewPort;
            env      : Port
    ): Typescript;
```

Abstract:

Creates a typescript inside a viewport. The typescript uses the system font, displays long lines by wrapping around, and stores three windows' worth of output.

Parameters:

ServPort	The master Typescript service port (TypescriptPort)
vp	Viewport to contain the typescript
env	Environment Manager connection, used to define the environment for escape completion on this typescript

Returns:

Port for a new typescript

3.2.3. Creating typescript in viewport with specified parameters

```
Function STSFullOpen(
            ServPort   : Port;
            vp         : ViewPort;
            env        : Port;
            fontName   : TString255;
            doWrap     : Boolean;
            dispPages  : Integer
    ): Typescript;
```

Abstract:

Creates a typescript inside a viewport, allowing the program to select more parameters for the typescript.

Parameters:

ServPort	The master Typescript service port (TypescriptPort)
vp	Viewport to contain the typescript
env	Environment Manager connection, used to define the environment for Escape Completion on this typescript
fontName	Name of the file containing the font to use for the typescript. It must be an absolute path name
doWrap	Always true
dispPages	Number of screens' worth of typescript to save

Returns:

Port for a new typescript

3.2.4. Creating typescript in viewport with cursor addressing

```
Function STSOpenTerm(
            ServPort    : Port;
            vp          : Viewport;
            w           : Port;
            env         : Port;
            fontname    : TString255;
            dispPages   : Integer;
            keytrantab  : pKeyTab;
            tablesize   : Long
        ): Typescript;
```

Abstract:

Creates a typescript inside a viewport with cursor addressing.

Parameters:

ServPort	TypescriptPort
vp	Viewport to display output on
w	Window to get input on. If null, client process will handle input
env	Environment Manager connection for escape completion
fontname	Absolute pathname of file containing font for typescript
dispPages	Number of pages of display to remember
keytrantab	Pointer to a key translation table that has been loaded
tablesize	Size of the key translation table in bytes

Returns:

Port to make typescript requests on

3.2.5. Creating typescript in window with defaults

```
Function STSOpenWindow(
            ServPort : Port;
            w        : Window;
            env      : Port
        ): Typescript;
```

Abstract:

Creates a typescript inside a window. The typescript uses the system font, displays long lines by wrapping around, and stores three windows' worth of output.

Parameters:

ServPort	TypescriptPort
w	Window to contain the typescript
env	Environment Manager connection for escape completion

Returns:

Port for a new typescript

3.2.6. Creating typescript in window, with specified parameters

```
Function STSFullOpenWindow(
            ServPort   : Port;
            w          : Window;
            env        : Port;
            fontName   : TString255;
            doWrap     : Boolean;
            dispPages  : Integer
    ): Typescript;
```

Abstract:

Creates a typescript inside a window, allowing the program to select more parameters for the typescript.

Parameters:

ServPort	TypescriptPort
w	Window to contain the typescript
env	Environment Manager connection for Escape Completion
fontName	Absolute pathname of the file containing the font to use for the typescript
doWrap	Always true
DispPages	Number of screens' worth of typescript to save

Returns:

Port for a new typescript

3.2.7. Reading an input character

```
Function STSGetChar(
            ServPort : Typescript
        ): Char;
```

Abstract:

Returns a single character of input from a typescript. If a full line has not been typed, it invokes the line editor and waits until a line is completed. If a line has been typed, it removes the next character from the start of the line and returns it.

Parameters:

ServPort	Port for the typescript

Returns:

The first character on the input line

3.2.8. Reading a line of input

```
Function STSGetString(
            ServPort : Typescript
        ): TString255;
```

Abstract:

Returns an entire line of input from a typescript. If a full line has not been typed, it invokes the line editor and waits until a line is completed. It then returns the entire line.

Parameters:

ServPort Port for the typescript

Returns:

The entire input line

3.2.9. Writing a single character

```
Procedure STSPutChar(
            ServPort : Typescript;
            ch       : Char
        );
```

Abstract:

Writes a single character to a typescript.

A LineFeed character ends the current output line. A BELL character (control G) flashes the viewport containing the typescript. Any other character is appended to the current line, possibly translated:

Control characters (chr(0)..chr(31)) are displayed as ^Char.

Printing characters (space through '}') are displayed as themselves.

DEL is displayed as ^{.

Any character greater than chr(127) is displayed as the corresponding

character in the font for the typescript, minus 128. This allows characters in the font with numeric values less than 32 to be displayed as normal printing characters.

Parameters:

ServPort	Port for the typescript
Ch	Single character to output

3.2.10. Writing a string of characters

```
Procedure STSPutString(
              ServPort : Typescript;
              s        : TString255
        );
```

Abstract:

Writes a string of characters to a typescript. Each character in the string is displayed according to the description for STSPutChar.

Parameters:

ServPort	Port for the typescript
s	String to output

3.2.11. Writing a character array

```
Procedure STSPutCharArray(
            ServPort   : Port;
            chars      : pTSCharArray;
            char_count : Long;
            firstCh    : Integer;
            lastCh     : Integer
           );
```

Abstract:

Writes a character array to a typescript. Each character in the array is displayed according to the description for STSPutChar.

Parameters:

ServPort	Port for the typescript
chars	Pointer to the character array
char_count	Number of characters in array

firstCh Position in array of first character to write

lastCh Position in array of last character to write

3.2.12. Setting input line

```
Function STSSetInput(
            ServPort : Port;
            inputstr : TString255
        ): Generalreturn;
```

Abstract:

Sets the current input line to be the input string given. This will put as much of the input string as will fit at the current location.

Parameters:

ServPort Port for the typescript

inputstr String to set the current input line to

Returns:

Success

Failure If the whole input string could not be inserted

3.2.13. Flushing partially read input

```
Procedure STSFlushInput(
            ServPort     : Typescript
        );
```

Abstract:

Flushes any partially entered input line from a typescript.

Parameters:

ServPort Port for the typescript

3.2.14. Flushing queued output

```
Procedure STSFlushOutput(
            ServPort : Typescript
        );
```

Abstract:

Forces any queued output for a typescript to be displayed on the screen. STSPutChar and STSPutString may not display the output characters immediately, giving strange results if the same viewport is used for simple text output and for graphics. STSFlushOutput ensures that all typescript output is displayed on the screen before it returns.

Parameters:

ServPort Port for the typescript

3.2.15. Waiting for full input line

```
Function STSFullLine(
                ServPort : Port
        ): Boolean;
```

Abstract:

Returns True if the input line is a full line (if it has been completed with a Return). STSFullLine will not return any value until the input line has been completed; therefore this routine will never return False.

Parameters:

ServPort Port for the typescript

Returns:

True

3.2.16. Checking input line status

```
Procedure STSInputStatus(
                ServPort : Port;
            var empty    : boolean;
            var full     : boolean
        );
```

Abstract:

Returns the status of the input line (full, empty, or neither).

Parameters:

ServPort	Port for the typescript
empty	Returns true if input line is empty
full	Returns true if input line is full

3.2.17. Finding dimensions of viewport

```
Procedure STSVpSize(
                ServPort  : Port;
            var charwidth : Integer;
            var lines     : Integer;
            var pixwidth  : Integer;
            var pixheight : Integer
        );
```

Abstract:

Returns the width and height of the viewport in characters and lines, and in pixels.

Parameters:

ServPort	Port for the typescript
charwidth	Width of viewport in characters
lines	Number of lines that will fit in viewport
pixwidth	Width of viewport in pixels
pixheight	Height of viewport in pixels

3.2.18. Finding position of cursor in viewport

```
Function STSCursorPos(
                ServPort : Port;
            var charpos  : Integer;
            var linenum  : Integer;
            var x        : Integer;
            var y        : Integer
        ): GeneralReturn;
```

Abstract:

Returns the position of the cursor in a viewport by character position and line number and by coordinates in pixels. Note that the STSOpenTerm routine must be called on the typescript before this function can be called.

Parameters:

ServPort	Port for the typescript
charpos	Position of the cursor in characters from the left edge of the viewport (position at beginning of line is 1)
linenum	Number of the line the cursor is on (the top line of a viewport is 1)
x	x-coordinate of cursor in pixels
y	y-coordinate of cursor in pixels

Returns:

Success	
Failure	If typescript was not opened with STSOpenTerm

3.2.19. Changing the cursor position

```
Function STSMoveCursor(
                ServPort : Port;
            var charpos  : Integer;
            var linenum  : Integer
        ): GeneralReturn;
```

Abstract:

Move the cursor to a specified absolute position on a viewport and return the new position.

Parameters:

ServPort	Port for the typescript
charpos	Position of the cursor in characters from the left edge of the viewport
linenum	Number of the line containing the cursor

Returns:

Success	If the cursor is moved to the position specified
Failure	If cursor was not moved to the position specified because: character position <1 line number <1 line number > number of lines in viewport

3.2.20. Performing a cursor addressing operation

```
Function STSCursorOp(
                ServPort : Port;
                op       : CursorOp;
                count    : Integer;
            var charpos  : Integer;
            var linenum  : Integer
        ): GeneralReturn;
```

Abstract:

Performs the specified cursor addressing operation and returns the new position of the cursor. Note that STSOpenTerm must be called on the typescript before this routine can be used.

Parameters:

ServPort	Port for the typescript	
op	Cursor addressing operation to perform. One of:	
	ts_home	move cursor to home position (upper left-hand corner)
	ts_boln	move cursor to beginning of present line
	ts_eoln	move cursor to end of present line
	ts_up	move cursor up one line
	ts_down	move cursor down one line
	ts_left	move cursor one character position to the left
	ts_right	move cursor one position to the right
count	Number of times to do the operation	
charpos	Position of the cursor in characters from the left edge of the viewport	
linenum	Number of the line containing the cursor	

Returns:

Success	If the cursor operation succeeded the specified number of times
Failure	If the operation failed to execute the specified number of times or an illegal cursor operation was given

3.2.21. Performing a delete operation

```
Function STSDeleteOp(
                ServPort : Port;
                op       : DeleteOp;
                count    : Integer;
            var charpos  : Integer;
            var linenum  : Integer
        ): GeneralReturn;
```

Abstract:

Performs the specified deletion and returns the new position of the cursor. Note that STSOpenTerm must be called on the typescript before this routine can be used.

Parameters:

ServPort	Port for the typescript
op	Delete operation to perform. One of:
	ts_delchar delete character at position of cursor
	ts_ereoln delete from cursor position to end of line
	ts_erboln delete from cursor position to beginning of line
count	Number of times to do the operation
charpos	Position of the cursor in characters from the left edge of the viewport
linenum	Number of the line containing the cursor

Returns:

Success	If the delete operation succeeds the specified number of times
Failure	If the operation failed to execute the specified number of times or an illegal delete operation was given

3.2.22. Performing a screen operation

```
Function STSScreenOp(
                ServPort    : Port;
                op          : ScreenOp;
                topline     : Integer;
                bottomline  : Integer;
                count       : Integer
        ): GeneralReturn;
```

Abstract:

Performs the specified screen operation on a subviewport. Note that STSOpenTerm must be called on the typescript before this routine can be used.

Parameters:

ServPort	Port for the typescript
op	Screen operation to perform. One of:
	ts_bell flash the viewport
	ts_clear clear the viewport
	ts_redisplay redisplay the contents of the viewport
	ts_scroll scroll the typescript in the viewport
topline	Line number indicating top of subviewport to do operation on
bottomline	Line number indicating bottom of subviewport to do operation on
count	Number of times to do the operation

Returns:

Success	If the operation succeeded the specified number of times
Failure	If the operation failed to execute the specified number of times, illegal operation given, or both line numbers not >0.

3.2.23. Changing the environment

```
Procedure STSChangeEnv(
            ServPort    : Typescript;
            env         : Port
        );
```

Abstract:

Changes the Environment Manager connection associated with a typescript. The Environment Manager connection determines the searchlists used for

escape completion within that typescript.

Parameters:

ServPort Port for the typescript

env New Environment Manager connection to use

3.2.24. Taking control of a window

```
Function STSGrabWindow(
            ServPort     : Typescript;
            kPort        : Port
        ): Window;
```

Abstract:

STSGrabWindow tells the Typescript Manager to stop monitoring the state of the window containing a typescript for change in state, size, or coveredness. A program such as the Editor uses this procedure to gain control of the default user window to use it for graphics. When the program terminates, the Typescript Manager regains control of the window and redisplays its contents as of the time STSGrabWindow was called; any changes that the program made to the window are lost.

Parameters:

ServPort Port for the typescript

kPort Port that the user program has ownership rights for or that will otherwise be deallocated when the program terminates. The Typescript Manager regains control of the window when this port is deleted.

Returns:

The window that the typescript is using

3.2.25. Passing an untranslated key

```
Function STSGiveKey(
                ServPort   : Port;
                untrankey  : EventRec;
            var fullLine   : boolean
        ): GeneralReturn;
```

Abstract:

Gives an untranslated key to the Typescript Manager as input. If typescript is blocked in more mode, the typescript will be unblocked and the key will be ignored.

Parameters:

ServPort	Port for the typescript to get the key	
untrankey	Untranslated key event	
fullLine	One of:	
	True	key is a carriage return
	False	key is not a carriage return

Returns:

Success	
Failure	If no key translation table for this typescript, or if key could not be translated

3.2.26. Changing a key translation table

```
Function STSChangeKeyTran(
            ServPort    : Port;
            keytrantab  : pKeyTab;
            tablesize   : Long
        ): GeneralReturn;
```

Abstract:

Changes the key translation table used by the specified typescript.

Parameters:

ServPort	Port for the typescript
keytrantab	Pointer to a loaded key translation table to be used
tablesize	Size of the table in bytes

Returns:

Success

Failure If new key translation table not valid

3.2.27. Turning More mode on or off

```
Function STSMoreMode(
                ServPort : Port;
             var on      : boolean
        ): GeneralReturn;
```

Abstract:

Turns More mode on or off. When More mode is off, the typescript is in Continual Scroll mode.

Parameters:

ServPort Port for the typescript

on One of:

True turn more mode on

False turn more mode off

Is set to whether more mode was on or off before this call

4. Input Editing Commands

The following commands can be used to edit input. These commands are also described in the "Basic Operations" document in the *Accent User's Manual*.

CTRL a	= beginning of line
CTRL e	= end of line
CTRL f	= forward character
CTRL b	= backward character
CTRL d	= delete character forward
CTRL h, DEL	= delete character backward
BackSpace	= character backward
CTRL k	= kill to end of line
CTRL u, OOPS	= kill to beginning of line
Tab	= go to next tab position (every 8 characters)
CTRL t	= Exchange characters before cursor
RETURN	= send off line
CTRL l	= refresh typescript
CTRL D	= delete word forward
CTRL H	= delete word backward
CTRL F	= forward word
CTRL B	= backward word
CTRL v	= display next page of output
CTRL V	= display previous page of output
CTRL p	= retrieve previous command in ring buffer
CTRL n	= retrieve next command in ring buffer
LF	= unblock window (next screenful of text)
CTRL LF	= set 'more' mode on
CTRL \	= set 'more' mode off
CTRL ?	= expand wild path
INS	= complete wild path