Cortland Text Tools

May 18, 1986

Revision History

March 10, 1986	Ver. 0.11	R. Montagne	Initial release. (note that this toolset was created from the Text, Basic and Pascal functions previously implemented in the Miscellaneous Tools).
March 31, 1986	Ver. 0.12	R. Montagne	**THIS IS A MAJOR REVISION!!!** The text tool set has been redefined in order to make the tool set more usable. One call type supports BASIC, PASCAL and RAM based drivers. Global parameters have changed. READ IT ALL!!! **This will be implemented in the BETA 2.0 ROMS!!!**
April 3, 1986	Ver. 0.13	R. Montagne	Added Error Device Global Masks and get I/O directing information.
April 23, 1986	Ver. 0.14	R. Montagne	Changed concatenation of characters in WrLine for RAM based drivers only. THIS IS THE BETA 2.0 IMPLEMENTATION.
May 18, 1986	Ver. 0.15	R. Montagne	Corrected errors in GetOutGlobals & GetErrGlobals. Added note in Directing I/O functions with regard to Apple][I/O hooks (italics).

Text Tools. The text tool set provides an interface between Apple][character device drivers which must be executed in emulation mode, and new applications running in native mode. It also provides a means of redirection of I/O through ram based drivers. The Text Tools (Tool set number = $0C) make it possible to deal with the text screen without switching modes and moving to bank zero. Dispatches to Ram based drivers will occur in full native mode (16 bit 'm' and 'x').

Standard Tool Set Calls.

TextBootInit Function number = $01

This function sets up the default device parameters as follows:

1.	Input device type is BASIC
2.	Output device type is BASIC
3.	Error output device type is BASIC
4.	Input device resides in slot #3
5.	Output device resides in slot #3
6.	Error output device resides in slot #3
7.	Global input AND mask is set to $FF
8.	Global input OR mask is set to $80
7.	Global output AND mask is set to $FF
8.	Global output OR mask is set to $80
7.	Global error output AND mask is set to $FF
8.	Global error output OR mask is set to $80

TextStartUp Function number = $02

This does nothing.

TextShutDown Function number = $03

This does nothing.

TextVersion Function number = $04

	Input	Word	Space for result
sp—>			
	Output	Word	Version number
sp—>			

This tool returns the version number of the Text Tool Set.

TextReset Function number = $05

This function sets up the default device parameters as follows:

1.	Input device type is BASIC
2.	Output device type is BASIC
3.	Error output device type is BASIC
4.	Input device resides in slot #3
5.	Output device resides in slot #3
6.	Error output device resides in slot #3
7.	Global input AND mask is set to $FF
8.	Global input OR mask is set to $80
7.	Global output AND mask is set to $FF
8.	Global output OR mask is set to $80
7.	Global error output AND mask is set to $FF
8.	Global error output OR mask is set to $80

TextStatus Function number = $06

	Input	Word	Space for result
sp—>			
	Output	Word	Status ($0000=Inactive, $FFFF=Active)
sp—>			

This tool returns a status that indicates that the Text Tool Set is active.

TextSpare1 Function number = $07

This does nothing.

TextSpare2 Function number = $08

This does nothing.

Text Global Functions. These tools are used to set or read the current global parameters used by the Pascal and Basic text tools. Characters are logically ANDed with the AND mask, and then logically ORed with the OR mask by the Pascal and Basic text tools.

```
SetInGlobals      Function number = $09

      Input       Word           AND mask
      Input       Word           OR mask
sp—>
```

Sets the global parameters for the input device.

```
SetOutGlobals     Function number = $0A

      Input       Word           AND mask
      Input       Word           OR mask
sp—>
```

Sets the global parameters for the output device.

```
SetErrGlobals     Function number = $0B

      Input       Word           AND mask
      Input       Word           OR mask
sp—>
```

Sets the global parameters for the error output device.

GetInGlobals Function number = $0C

```
        Input     Word        Space for result
        Input     Word        Space for result
sp—>

        Output    Word        AND mask
        Output    Word        OR mask
sp—>
```

Returns with the current values for the input device global parameters.

GetOutGlobals Function number = $0D

```
        Input     Word        Space for result
        Input     Word        Space for result
sp—>

        Output    Word        AND mask
        Output    Word        OR mask
sp—>
```

Returns with the current values for the Output device global parameters.

GetErrGlobals Function number = $0E

```
        Input     Word        Space for result
        Input     Word        Space for result
sp—>

        Output    Word        AND mask
        Output    Word        OR mask
sp—>
```

Returns with the current values for the Error Output device global parameters.

Directing I/O Functions. These tool functions are provided to direct I/O from the Text Tool Set to a specific type of character device driver or inquire information about the directing of a specific I/O driver. Three types of character device drivers are supported.

Device Type	Device Description
0	BASIC Device Driver
1	PASCAL Device Driver
2	RAM Based Device Driver
≥3	Illegal Driver Type

BASIC device drivers must support the standard Apple][BASIC device driver entry points (INIT, INPUT, and OUTPUT). *It should be noted that the BASIC devices use the Apple][I/O hooks ($36-$39). Any desk accessories using the text tool set BASIC device drivers should save and restore the global masks, device descriptors and the I/O hooks when entering and exiting the DA.*

PASCAL device drivers must support the standard Apple][Pascal 1.1 device driver entry points (INIT, READ, WRITE, and status). The optional Pascal 1.1 control entry point is supported by the text tool set, but does not neccessarily have to be supported by the device. Dispatches to optional Pascal driver entry points that are not supported by the device will return a 'NO DEVICE CONNECTED error.

RAM based device drivers must support five entry points. These are INIT, READ, WRITE, STATUS and CONTROL. Ram based drivers may be located at any address and in any bank. Entry points must be supported by the RAM based driver as follows:

RAMDRIVER Base Address	Initialization entry point
RAMDRIVER Base Address+3	Read entry point
RAMDRIVER Base Address+6	Write entry point
RAMDRIVER Base Address+9	Status entry point
RAMDRIVER Base Address+12	Control entry point

The text tool functions profided for directing the I/O to or from a specific device driver follows:

SetInputDevice Function number = $0F

Input	Word	Device Type
Input	LongWord	Pointer or Slot

sp—>

The device type specifies the type of driver installed as the input device. The longword pointer points to the slot containing the device driver in the case of the device type being either a BASIC or PASCAL device driver. If the device type is a RAM based device driver, then the longword pointer points to the INITIALIZATION entry point within the RAM based driver.

SetOutputDevice Function number = $10

Input	Word	Device Type
Input	LongWord	Pointer or Slot
sp—>		

The device type specifies the type of driver installed as the output device. The longword pointer points to the slot containing the device driver in the case of the device type being either a BASIC or PASCAL device driver. If the device type is a RAM based device driver, then the longword pointer points to the INITIALIZATION entry point within the RAM based driver.

SetErrorDevice Function number = $11

Input	Word	Device Type
Input	LongWord	Pointer or Slot
sp—>		

The device type specifies the type of driver installed as the error output device. The longword pointer points to the slot containing the device driver in the case of the device type being either a BASIC or PASCAL device driver. If the device type is a RAM based device driver, then the longword pointer points to the INITIALIZATION entry point within the RAM based driver.

GetInputDevice Function number = $12

Input	Word	Space for result
Input	LongWord	Space for result
sp—>		

Output	Word	Device Type
Output	LongWord	Pointer or Slot
sp—>		

The device type returned specifies the type of driver installed as the input device. The longword pointer points to the slot containing the device driver in the case of the device type being either a BASIC or PASCAL device driver. If the device type is a RAM based device driver, then the longword pointer points to the INITIALIZATION entry point within the RAM based driver.

GetOutputDevice Function number = $13

	Input	Word	Space for result
	Input	LongWord	Space for result
sp—>			
	Output	Word	Device Type
	Output	LongWord	Pointer or Slot
sp—>			

The device type returned specifies the type of driver installed as the output device. The longword pointer points to the slot containing the device driver in the case of the device type being either a BASIC or PASCAL device driver. If the device type is a RAM based device driver, then the longword pointer points to the INITIALIZATION entry point within the RAM based driver.

GetErrorDevice Function number = $14

	Input	Word	Space for result
	Input	LongWord	Space for result
sp—>			
	Output	Word	Device Type
	Output	LongWord	Pointer or Slot
sp—>			

The device type specifies the type of driver installed as the error output device. The longword pointer points to the slot containing the device driver in the case of the device type being either a BASIC or PASCAL device driver. If the device type is a RAM based device driver, then the longword pointer points to the INITIALIZATION entry point within the RAM based driver.

Text Functions. The tools specified below are provided to interface with any BASIC, PASCAL 1.1, or RAM based character device driver. Included are tool functions for initialization, control, input, output and status.

InitTextDev Function number = $15

Input	Word	Device to initialize

sp—>

Initializes the text device specified by the tool input parameters as follows:

Parameter	Device
0	Input Device
1	Output Device
2	Error Output Device
≥3	Illegal parameter value

CtrlTextDev Function number = $16

Input	Word	Device to control
Input	Word	Control code (in low byte)

sp—>

Passes the control code to the text device specified by the tool input parameters.

Parameter	Device
0	Input Device
1	Output Device
2	Error Output Device
≥3	Illegal parameter value

Basic devices do not support this function. The text tool will return an error if this call is made to a basic device. Note that for PASCAL device drivers, this is an optional entry point and may not be supported by all Pascal devices.

StatusTDev Function number = $17

Input	Word	Device to request status from
Input	Word	Request code (in low byte)

sp—>

Executes a status call to the text device specified by the tool input parameters.

Parameter	Device
0	Input Device
1	Output Device
2	Error Output Device
≥3	Illegal parameter value

WriteChar Function number = $18

Input	Word	Character (in low byte of word)
sp—>		

The character is combined with the output global AND mask and OR mask, and then is written to the text device specified by the output device.

ErrWriteChar Function number = $19

Input	Word	Character (in low byte of word)
sp—>		

The character is combined with the error output global AND mask and OR mask, and then is written to the text device specified by the error output device.

WriteLine Function number = $1A

Input	LongWord	Pointer to ASCII string
sp—>		

The character string with a length specified by the first byte in the string is combined with the ouput global masks, and is then written to the text device specified as the output device. For BASIC and RAM based drivers, a carriage return will be concatenated to the string by the tool. For PASCAL drivers, a carriage return and line feed will be concatenated to the string by the tool.

ErrWriteLine Function number = $1B

Input	LongWord	Pointer to ASCII string
sp—>		

The character string with a length specified by the first byte in the string is combined with the error ouput global masks, and is then written to the text device specified as the error output device. For BASIC and RAM based drivers, a carriage return will be concatenated to the string by the tool. For PASCAL drivers, a carriage return and line feed will be concatenated to the string by the tool.

WriteString Function number = $1C

Input	LongWord	Pointer to ASCII string
sp—>		

The character string with a length specified by the first byte in the string is combined with the output global masks, and is then written to the text device specified as the output device.

ErrWriteString Function number = $1D

Input	LongWord	Pointer to ASCII string
sp—>		

The character string with a length specified by the first byte in the string is combined with the error output global masks, and is then written to the text device specified as the error output device.

WriteBlock Function number = $1E

Input	LongWord	Pointer to ASCII text
Input	Word	Offset
Input	Word	Count
sp—>		

The character string with a length specified by the **Count** at the memory location **Pointer+Offset** is combined with the output global masks, and is then written to the text device specified as the output device.

ErrWriteBlock Function number = $1F

Input	LongWord	Pointer to ASCII text
Input	Word	Offset
Input	Word	Count
sp—>		

The character string with a length specified by the **Count** at the memory location **Pointer+Offset** is combined with the error output global masks, and is then written to the text device specified as the error output device.

WriteCString Function number = $20

Input	LongWord	Pointer to ASCII C-String
sp—>		

The character string terminating with the byte value of $00 is combined with the output global masks, and is then written to the text device specified as the output device.

ErrWriteCString Function number = $21

Input	LongWord	Pointer to ASCII C-String
sp—>		

The character string terminating with the byte value of $00 is combined with the error output global masks, and is then written to the text device specified as the error output device.

ReadChar Function number = $22

Input	Word	Space for result
Input	Word	Echo Flag
sp—>		
Output	Word	Character (in low byte)
sp—>		

The character read from the text device that has been set as the input device is combined with the input global masks and returned on the stack. If the **ECHO** flag is set to a value of $0001, then the character read from the input device will be written to the output device. If the **ECHO** flag is set to zero, then the character will not be written to the output device.

ReadBlock Function number = $23

Input	LongWord	Pointer
Input	Word	Offset
Input	Word	BlockSize
Input	Word	Echo Flag
sp—>		

The block of characters of the size specified by **BlockSize** is read from the text device that has been set as the input device, and is combined with the input global masks before being written to the memory location specified by **Pointer+Offset**. If the **ECHO** flag is set to a value of $0001, then the character read from the input device will be written to the output device. If the **ECHO** flag is set to zero, then the character will not be written to the output device.

ReadLine Function number = $24

Input	Word	Space for result
Input	LongWord	BufferPointer
Input	Word	MaxCount (maximum line length)
Input	Word	EOL (end of line character in low byte)
Input	Word	Echo Flag
sp—>		
Output	Word	Count of characters received.
sp—>		

The character string is read from the text device that has been set as the input device, and is combined with the input global masks before being written to the memory location specified by **BufferPointer**. The character string is terminated by an **EOL** character, or if the count of characters received is equal to the maximum line length specified by **MaxCount**. The count of characters received is returned on the stack. If the **ECHO** flag is set to a value of $0001, then the character read from the input device will be written to the output device. If the **ECHO** flag is set to zero, then the character will not be written to the output device.

Summary of functions within the Text Tool Set:

Function Number		Function Description
$01	1	TextBootInit
$02	2	TextStartUp
$03	3	TextShutDown
$04	4	TextVersion
$05	5	TextReset
$06	6	TextStatus
$07	7	TextSpare1
$08	8	TextSpare2
$09	9	SetInGlobals
$0A	10	SetOutGlobals
$0B	11	SetErrGlobals
$0C	12	GetInGlobals
$0D	13	GetOutGlobals
$0E	14	GetErrGlobals
$0F	15	SetInputDevice
$10	16	SetOutputDevice
$11	17	SetErrorDevice
$12	18	GetInputDevice
$13	19	GetOutputDevice
$14	20	GetErrorDevice
$15	21	InitTextDev
$16	22	CtrlTextDev
$17	23	StatusTextDev
$18	24	WriteChar
$19	25	ErrWriteChar
$1A	26	WriteLine
$1B	27	ErrWriteLine
$1C	28	WriteString
$1D	29	ErrWriteString
$1E	30	WriteBlock
$1F	31	ErrWriteBlock
$20	32	WriteCString
$21	33	ErrWriteCString
$22	34	ReadChar
$23	35	ReadBlock
$24	36	ReadLine