Appendix A

tiny-c Shell Commands

Command	Name	Arguments	Description
c	change	/string-1/string-2/	**Action:** Replace string-1 with string-2 in the current line; the slash (/) can be any character not appearing string-1 or string-2. **Default:** If no arguments are given, the previous change command is repeated.
d	delete	n	**Action:** Delete n lines, starting at the current line. **Default:** If not given, n is taken to be 1.
l	locate	/string/	**Action:** The program text is searched forward starting at the current line for the first appearance of string; the slash (/) may be any character

Command	Name	Arguments	Description
			not appearing in string. **Default:** If no argument is given, the previous locate command is repeated.
p	print	n	**Action: Print** n lines starting with the current line. **Default:** If not given, n is taken to be 1.
r	read	filename	**Action:** The contents of the file named filename are appended to the end of the program text.
w	write	filename	**Action:** The contents of the program text area are written to the file named file name.
x	exit	(none)	**Action:** tiny-c exits and returns to DOS.
+	next	n	**Action:** The current line number isincremented by n. **Default:** If not given, n is taken to be 1.
–	last	n	**Action:** The current line number is decremented by n. **Default:** If not given, n is taken to be 1.

Command	Name	Arguments	Description
/	status	(none)	**Action:** Prints the current line number, the total number of lines, the total number of characters used, and the total number of characters unused.

Appendix B

tiny-c Error Codes

1 Illegal statement
2 Cursor ran off end of program. Look for missing] or).
3 Symbol error. A name was expected. For example, 10 + + will cause this.
5 Right parenthesis missing, as in x = (x + b*b.
6 Subscript out of range.
7 Using a pointer as a variable or vice versa.
9 More expression expected, as in x = x +.
14 Illegal equal sign, as in 7=2
16 Stack overflow. Either an expression is too tough or you are too deeply nested in functions or a recursion has gone too deep.
17 Too many active functions.
18 Too many active variables.
19 Too many active values. Values share space with program text. Squeeze the program and this error may go away.
20 Startup error. Caused by a "garbage" line outside of all brackets [ ], i.e., where globals are declared. A missing [or] can cause this.
21 Arguments needed and number given don't agree.
22 A function body must begin with [.
24 An illegal invocation of MC.
26 Undefined symbol. Perhaps a variable name is misspelled, or you need an int or char statement for it, or a function isn't loaded.
99 Program interrupted.

Appendix C

The tiny-c System Library

The system library includes functions that do input, output, and character manipulation. The definitions given here show the declaration of the function name and the arguments, if any.

gs char buffer(0)	Reads a line, i.e., a string of characters terminated by a carriage return, from the terminal and puts it in buffer. The carriage return at the end of the line is changed to a null byte. The value of the function is the number of characters placed in buffer excluding the null byte. A value of 0 is permitted.
ps char buffer (0)	Prints the string in buffer on the screen. A null byte signals the end of the string. The null is not transmitted. The number of characters transmitted is returned as the value of the function.
pl char buffer(0)	The same as ps but prints the string on a new line. The number of characters transmitted, not including the leading return and line feed, is returned.
pn int n	Prints the value of n on the screen preceded by a blank.

	The number of characters transmitted, including the blank, is returned.
gn	Reads a line and returns the integer at the beginning of the line. If there is no integer there, it prints "number required" and tries again.
gc	Reads a line and returns the first character on the line.
putchar char c	Transmits the character **c** to the terminal. Any character, including control characters, can be transmitted, except that a quote is transmitted if **c** is null. The character **c** is returned.
getchar	Reads and returns a character from the terminal. Any character, including control characters, can be read by this function.
readfile char name (0), where (0), limit (0) **int unit**	Reads data from a file. The **name** argument is a character string terminated by a null byte; **where** and **limit** are pointers. The **unit** designator is an input/output unit number. The file with name **name** is opened for reading on device **unit**. All of its records are read and placed in sequentially higher addresses, starting at **where** but in no case going beyond **limit**. Then **unit** is closed. If successful, the total number of bytes read is returned. If **limit** is exceeded, the message "too big" is printed and −2 is returned.
writefile char name(0), from(0), to(0) **int unit**	Writes data to a file. The **name** argument is a character string terminated by a null byte; **from** and **to** are pointers. The **unit** argument is an input/output unit number. Function **writefile** opens unit **unit** for output. The contents of sequentially higher addresses from **from** to **to** inclusive are written to **unit** as a file named **name**. Then the file is closed. If successful, the total number of bytes written is returned. If a problem occurs, a negative value is returned.
num char b(5) **int v(0)**	Converts a string of digits without leading sign or blanks

	to the corresponding numeric value, which is put in **v(0)**. The first non-digit stops the conversion. At most, five digits are examined. The number of bytes converted is returned as the value of the function. Note that the second argument must be a pointer to an integer.
atoi char b(0) **int v(0)**	Converts a character string of this form: 0 or more blanks optional + or − character 0 or more blanks 0 to 5 digits to its numeric value, which is put in **v(0)**. The first non-digit following the digit part stops the conversion. The number of characters in **b** that were used to form the value is returned as the value of the function.
ceqn char a(0), b(0) **int n**	Compares two character strings for equality for **n** characters. Returns 1 on equals, 0 on not-equals.
alpha char c	Returns a 1 if **c** is an alphabetic character, upper- or lowercase. Otherwise returns a 0.
index char s1(0) **int n1** **char s2(0)** **int n2**	Finds the leftmost copy of the character string **s1**, which is **n1** bytes long in the character string **s2**, which is **n2** bytes long. If **s1** does not appear in **s2**, 0 is returned. If **s1** does appear, returns **n+1** such that **s2+n** points to the first character of the copy in **s2**.
move char a(0), b(0)	Moves string **a** into **b** up to and including the null byte of **a**.
movebl char a(0), b(0) **int k**	Moves a block of storage up or down **k** bytes in memory; **a** and **b** point to the first and last characters of the block to be moved. If **k** is positive, the move is to higher addresses, and if it is negative the move is to lower addresses. If **k** is positive, the byte at **b** is moved first, then the byte at **b−1**, etc. If **k** is negative, the byte at **a** is moved first. Large blocks thus can be moved a few bytes without destruction.
countch char a(0), **b(0), c**	Counts the instances of the character **c** in the block of storage from **a** to **b** inclusive and returns the count.

scann char from(0), to (0), c int n(0)	Scans from **from** to **to** inclusive for instances of the character **c**. The integer **n(0)** is decremented for each **c** found. If **n(0)** reaches 0, or if the character in **to(0)** is examined, **scann** stops. Function **scann** returns the offset relative to the pointer **from** to the last examined character. Thus, if the third character position after **from** is the last examined, **scann** returns 3.
chrdy	Returns a copy of an input character from the terminal if a character has been typed but not yet read by another function, except that if the typed character is a null, a 1 is returned. If no unread character has been typed, a null byte is returned. The character is not cleared so a subsequent call to **getchar** or **gc** will return the same character.
pft char a(0), b(0)	Transfers all characters from **a** to **b** inclusive to the screen.
fopen int rw char name(0) int size, unit	Opens or creates a file for access on logical unit number **unit**. The **name** variable contains a null-terminated string giving the name of the file. The file is opened for reading if **rw** is 1, and writing if **rw** is 2. If **rw** is 2 and the file does not exist, then it will be created and its size guaranteed to be at least **size** bytes. Otherwise **size** is ignored, but it must be given. If no error is detected, a 0 is returned. If an error is detected a nonzero value is returned.
fread char a(0) int unit	Starting at **a**, reads into memory the next 512-byte record of data from the file opened on unit number **unit**. The array **a** must be large enough to hold an entire record. The length in bytes of the record actually placed at **a** is returned as the value of **fread**. A value of −1 is returned if an end-of-file is detected.
fwrite char from(0), to (0) int unit	Writes one record with the bytes from **from** to **to** inclusive to the file opened on unit number **unit**. This becomes the next record of the file. Its length is **to-from + 1**; this is the length that will be returned when the record is read by **fread**.

fclose int unit	The file opened on unit number unit is made permanent and arrangements are made for end-of-file detection by fread.
beep int f, d	The beeper is sounded at frequency f for d tenths of a second.

Appendix D

System and User Machine Calls for the IBM PC and PCjr

These machine calls are furnished with the tiny-c Interpreter. They are always loaded and available for use, and are called as described in Chapter 4. The function number is the *last* argument. Thus, MC 1 is called like this:

```
MC ('x', 1)
```

Where no results are indicated, a zero is returned as the value of the MC.

Function:	MC(character, 1)
Arguments:	A character value.
Results:	None
Errors:	None
Action:	character is transmitted to the screen.

Function:	MC(2)
Arguments:	None
Results:	A character value.
Errors:	None
Action:	A character is retrieved from the console terminal and returned as the value of the function. The input port from the console terminal is cleared to receive another character.

Function:	MC(mode, name, size, channel, 3)
Arguments:	A mode, file name, file size, and channel number.
Results:	An open status indicator.
Errors:	None
Action:	Same as **fopen** in the standard library (see Appendix C).

Function:	MC (buffer, channel, 4)
Arguments:	Pointer and a channel number.
Results:	A get-status indicator.
Errors:	None
Action:	Same as **fread** in the standard library (see Appendix C).

Function:	MC (from, to, 5)
Arguments:	Two pointers and a channel number.
Results:	A put-status indicator.
Errors:	None
Action:	Same as **fwrite** in the standard library (see Appendix C).

Function:	MC (channel, 6)
Arguments:	A channel number.
Results:	None
Errors:	None
Action:	Same as **fclose** in the standard library (see Appendix C).

Function:	MC (from, to, bytes, 7)
Arguments:	Two pointers and an integer.
Results:	None
Errors:	None
Action:	Same as **moveb1** in the standard library (see Appendix C).

Function:	MC (from, to, character, 8)
Arguments:	Two pointers and a character value.
Results:	The number of appearances of the character between the two pointers, inclusively.
Errors:	None
Action:	Same as **countch** in the standard library (see Appendix C).

Function:	MC (from, to, character, nptr, 9)
Arguments:	Two pointers, a character value, and an integer pointer.
Results:	A pointer to the last character examined.
Errors:	None
Action:	Same as **scann** in the standard library (see Appendix C).

Function:	MC (10)
Arguments:	None
Results:	None
Errors:	None
Action:	An immediate return to the operating system.

Function:	MC (facts, start, first, last, 11)
Arguments:	Four pointers.
Results:	None
Errors:	None
Action:	See Chapter 4.

Function:	MC (12)
Arguments:	None
Results:	A character value.
Errors:	None
Action:	Same as chrdy in the standard library (see Appendix C).

Function:	MC (from, to, 13)
Arguments:	Two pointers.
Results:	None
Errors:	None
Action:	Same as pft in the standard library (see Appendix C).

Function:	MC (n, 14)
Arguments:	An integer.
Results:	None
Errors:	None
Action:	Same as pn in the standard library (see Appendix C).

Function:	MC (buffer, 15)
Arguments:	A character buffer.
Results:	None
Errors:	None
Action:	Same as gs in the standard library (see Appendix C).

Function:	MC (start, bytes, character, 16)
Arguments:	A pointer, a number of bytes, a character.
Results:	None
Errors:	None
Action:	The argument number of bytes starting at start are set to character.

Function:	MC (frequency, duration, 17)
Arguments:	Two integers.
Results:	None
Errors:	None
Action:	Same as beep in the standard library (see Appendix C).

In addition to the seventeen standard machine calls, the IBM PC and PCjr version of tiny-c comes with three user machine calls. They are included primarily to demonstrate the use of user machine calls. They implement an interface to the sound system of the PCjr.

Function: MC (noise_type, attenuation, 1000)
Arguments: Two integers.
Results: **None**
Errors: None
Action: The noise generator is turned on. The noise_type is coded as follows:

0	periodic noise shifted at 6992 Hz
1	periodic noise shifted at 3496 Hz
2	periodic noise shifted at 1748 Hz
3	periodic noise shifted by tone voice 2
4	white noise shifted at 6992 Hz
5	white noise shifted at 3496 Hz
6	white noise shifted at 1748 Hz
7	white noise shifted by tone voice 2

and the attenuation is coded as follows:

0	OFF
1	28 dB
2	26 dB
3	24 dB
4	22 dB
5	20 dB
6	18 dB
7	16 dB
8	14 dB
9	12 dB
10	10 dB
11	8 dB
12	6 dB
13	4 dB
14	2 dB

Function: MC (voice, frequency, attenuation, 1001)

Arguments:	Three integers.
Results:	None
Errors:	None
Action:	Tone generator voice is turned on at the frequency and attenuation. Attenuation is coded as above and voice is 0, 1, or 2.

Function:	MC (frequency, 1002)
Arguments:	An integer.
Results:	None
Errors:	None
Action:	The frequency of the most recently mentioned tone generator voice is changed to frequency.

Bibliography

Aho, Alfred, and Ullman, Jeffrey. *Principles of Compiler Design*. Reading, Mass.: Addison-Wesley, 1978.

Brown, Peter. *Writing Interactive Compilers and Interpreters*. New York: John Wiley and Sons, 1979.

BYTE Special Issue on The C Language. August 1983.

Feurzeig, Wally, et al. "Programming Languages as a Conceptual Framework for Teaching Mathematics." *BBN Report* No. 2165, June 1971.

Duncan, Ray. "tiny-c Interpreter on CDOS." *Dr. Dobb's Journal*. May 1979.

Gibson, Tom. "Caution: Structured Programming Can Be Habit-Forming." *Creative Computing*, January 1979.

Gibson, Tom, and Guthery, Scott. "Structured Programming, C and tiny-c." *Dr. Dobb's Journal*, May 1980.

Gilbreath, Jim. "A High-Level Language Benchmark." *BYTE*, September 1981.

Gries, David. "On Structured Programming—A Reply to Smoliar." *CACM*, November 1974.

Hancock, Les. "Growing, Pruning and Climbing Binary Trees with tiny-c." *Dr. Dobb's Journal*, June/July 1979.

Hancock, Les, and Krieger, Morris. *The C Primer*. New York: McGraw-Hill, 1983.

Hughes, Phillip. "BASIC, Pascal or tiny-c." *BYTE*, October 1981.

Kemeny, J.G., and Kurtz, T.E. *BASIC Programming*. New York: John Wiley and Sons, 1967.

Kern, Christopher. "A User's Look at tiny-c." *Byte*, December 1979.

Kernighan, Brian, and Plauger, P. J. *Software Tools.* Addison-Wesley, 1967.

Kernighan, Brian, and Ritchie, Dennis. The C Programming Language. Englewood Cliffs, N.J.: Prentice-Hall, 1978.

Kernighan, Brian, and Pike, Rob. The UNIX Programming Environment. Englewood Cliffs, N.J.: Prentice-Hall, 1984.

Madden, J. Gregory, "C: A Language for Microprocessors?" *BYTE*, October 1977.

McIntire, Thomas. *Software Interpreters for Microcomputers*. New York: John Wiley and Sons, 1978.

Ritchie, Dennis and Thompson, Ken. "The UNIX Time-Sharing System." *CACM*, March 1974.

Ritchie, Dennis, et al. "The C Programming Language." Murray Hill, N.J.: Bell Telephone Laboratories, October 1975.

Ritchie, Dennis; Johnson, Steve; Lesk, Michael; and Kernighan, Brian. "The C Programming Language." *Dr. Dobbs Journal*, May 1980.

Sargent, Murray, and Shoemaker, Richard. *The IBM Personal Computer from the Inside Out*. Reading, Mass.: Addison-Wesley, 1984.

Snyder, Alan. "A Portable Compiler for the Language C." *MIT Project MAC Technical Report 149*, May 1975.

Wexelblat, Richard. "The Consequences of One's First Programming Language." *Software - Practice and Experience*, Vol. II, 1980.

Zahn, Carl. *Notes: A Guide to the C Programming Language*. New York: Yourdon Press, 1979.

Index