IMPORTANT PRODUCT INFORMATION FOR TI EXTENDED BASIC

TI Extended BASIC has been enhanced and modified for use with both the TI-99/4A and TI-99/4 Computers. Several important product differences should be noted in relation to the type of computer you have. Please read this folder and mark the appropriate changes in your copy of the *TI Extended BASIC* owner's manual.

Although the TI-99/4A and TI-99/4 Computers are similar, the TI-99/4A is easily recognizable by its standard typewriter keyboard which returns both upper-case (large capital) and lower-case (small capital) alphabetical characters. Depressing the **ALPHA LOCK** key locks the alphabet keys in upper-case mode. To release **ALPHA LOCK**, press the key again.

When the TI Extended BASIC module is in place, both the TI-99/4A and TI-99/4 Computers share several enhancements. However, each computer also has its own unique features. These features are discussed in the following paragraphs.

AUTO REPEAT FEATURE

When using TI Extended BASIC on either computer, holding down a key for more than one second automatically causes its symbol to be repeated on the display until you release the key.

TEXAS INSTRUMENTS
INCORPORATED
DALLAS, TEXAS

 1041361-3

SPECIAL FUNCTION KEYS

The TI-99/4A Computer has the same special computer functions as the TI-99/4. However, these functions are frequently assigned to different keys on the TI-99/4A Computer. The following chart compares the keystroke sequences for the function keys on the two units.

Function Keys

Key Name	*TI-99/4 Keys*	*TI-99/4A Keys*
AID	**SHIFT A**	**FCTN 7**
CLEAR	**SHIFT C**	**FCTN 4**
DELete	**SHIFT F**	**FCTN 1**
INSert	**SHIFT G**	**FCTN 2**
QUIT	**SHIFT Q**	**FCTN =**
REDO	**SHIFT R**	**FCTN 8**
ERASE	**SHIFT T**	**FCTN 3**
LEFT arrow	**SHIFT S**	**FCTN S**
RIGHT arrow	**SHIFT D**	**FCTN D**
DOWN arrow	**SHIFT X**	**FCTN X**
UP arrow	**SHIFT E**	**FCTN E**
PROC'D	**SHIFT V**	**FCTN 6**
BEGIN	**SHIFT W**	**FCTN 5**
BACK	**SHIFT Z**	**FCTN 9**
ENTER	**ENTER**	**ENTER**

In addition to these functions, the TI-99/4A Computer has functions represented as symbols on the fronts of the individual keyfaces. These functions may be accessed by pressing **FCTN** and the appropriate key simultaneously.

CONTROL KEYS

The TI-99/4A Computer also has control characters which are used primarily for telecommunications. To enter a control character, hold down the **CTRL** key and press the appropriate letter, number, or symbol key.

EXPANDED CHARACTER SET — TI-99/4A

As explained in your *TI Extended BASIC* manual, codes 32-95 are the predefined standard ASCII characters on the TI-99/4 Computer. The cursor and edge characters, ASCII codes 30 and 31, are assigned to character set 0. The undefined character codes (128-135 and 136-143) are assigned to sets 13 and 14, respectively.

These codes and the corresponding characters are listed in *Appendix C* of the manual. The CALL KEY character codes are also listed in *Appendix C*. *Appendix E* in the manual lists the 15 character code sets which may be used for color graphics.

Due to the inclusion of the lower-case character set, the defined characters on the TI-99/4A Computer are the standard ASCII characters for codes 32 through 127. The following chart lists these characters and their codes.

ASCII CODE	*CHARACTER*	*ASCII CODE*	*CHARACTER*
30	■ (cursor)	55	7
31	(edge character)	56	8
32	(space)	57	9
33	! (exclamation point)	58	: (colon)
34	" (quote)	59	; (semicolon)
35	# (number or pound sign)	60	< (less than)
36	$ (dollar)	61	= (equals)
37	% (percent)	62	> (greater than)
38	& (ampersand)	63	? (question mark)
39	' (apostrophe)	64	@ (at sign)
40	((open parenthesis)	65	A
41	) (close parenthesis)	66	B
42	* (asterisk)	67	C
43	+ (plus)	68	D
44	, (comma)	69	E
45	– (minus)	70	F
46	. (period)	71	G
47	/ (slant)	72	H
48	0	73	I
49	1	74	J
50	2	75	K
51	3	76	L
52	4	77	M
53	5	78	N
54	6	79	O

ASCII CODE	*CHARACTER*	*ASCII CODE*	*CHARACTER*
80	P	104	h } Displayed on screen as small capitals.
81	Q	105	i
82	R	106	j
83	S	107	k
84	T	108	l
85	U	109	m
86	V	110	n
87	W	111	o
88	X	112	p
89	Y	113	q
90	Z	114	r
91	[(open bracket)	115	s
92	\ (reverse slant)	116	t
93	] (close bracket)	117	u
94	^ (exponentiation)	118	v
95	_ (line)	119	w
96	` (grave)	120	x
97	a } Displayed on screen as small capitals.	121	y
98	b	122	z
99	c	123	{ (left brace)
100	d	124	| (vertical line)
101	e	125	} (right brace)
102	f	126	~ (tilde)
103	g	127	DEL (appears on screen as a blank)

CALL KEY SUBPROGRAM

The information given on the KEY subprogram in Chapter 4 of the *TI Extended BASIC* manual is accurate for the TI-99/4 Computer. The values of 3, 4, and 5 are not accessible as key units.

However, the TI-99/4A maps key units 0 through 5 to specific modes of operation. If the *key-unit* is 0, the keyboard is mapped in whichever mode was specified by the previous CALL KEY program line.

If the *key-unit* is 1, input is taken from the left side of the keyboard. If the *key-unit* is 2, input is taken from the right side of the keyboard.

A *key-unit* of 3 maps the computer into the standard TI-99/4 keyboard mode. Both upper- and lower-case characters are returned as upper-case characters only. Function codes 1 through 15 are active, but no control characters are returned.

A *key-unit* of 4 places the computer in Pascal mode with both upper- and lower-case characters active. The function codes 129 through 143 and the control character codes 1 through 31 are also active.

The *key-unit* 5 maps the TI-99/4A Computer in the BASIC mode. Both upper- and lower-case characters are active. The active function codes are 1 through 15, and the active control character codes are 128 through 159 (and 187).

In addition, codes are assigned to the function and control keys so that these can be referenced by the CALL KEY subprogram in TI Extended BASIC. The codes assigned depend on the key-unit value specified in a CALL KEY program statement. The following tables show typical code assignments.

FUNCTION KEY CODES

Codes			
TI-99/4 & BASIC Modes	*Pascal Mode*	*Function Name*	*Function Key*
1	129	**AID**	**FCTN 7**
2	130	**CLEAR**	**FCTN 4**
3	131	**DELete**	**FCTN 1**
4	132	**INSert**	**FCTN 2**
5	133	**QUIT**	**FCTN =**
6	134	**REDO**	**FCTN 8**
7	135	**ERASE**	**FCTN 3**
8	136	**LEFT arrow**	**FCTN S**
9	137	**RIGHT arrow**	**FCTN D**
10	138	**DOWN arrow**	**FCTN X**
11	139	**UP arrow**	**FCTN E**
12	140	**PROC'D**	**FCTN 6**
13	141	**ENTER**	**ENTER**
14	142	**BEGIN**	**FCTN 5**
15	143	**BACK**	**FCTN 9**

CONTROL KEY CODES

Codes				
BASIC Mode	*Pascal Mode*	*Mnemonic Code*	*Press*	*Comments*
129	1	SOH	**CONTROL A**	Start of heading
130	2	STX	**CONTROL B**	Start of text
131	3	ETX	**CONTROL C**	End of text
132	4	EOT	**CONTROL D**	End of transmission
133	5	ENQ	**CONTROL E**	Enquiry
134	6	ACK	**CONTROL F**	Acknowledge
135	7	BEL	**CONTROL G**	Bell
136	8	BS	**CONTROL H**	Backspace
137	9	HT	**CONTROL I**	Horizontal tabulation
138	10	LF	**CONTROL J**	Line feed
139	11	VT	**CONTROL K**	Vertical tabulation
140	12	FF	**CONTROL L**	Form feed
141	13	CR	**CONTROL M**	Carriage return
142	14	SO	**CONTROL N**	Shift out
143	15	SI	**CONTROL O**	Shift in
144	16	DLE	**CONTROL P**	Data link escape
145	17	DC1	**CONTROL Q**	Device control 1 (X-ON)
146	18	DC2	**CONTROL R**	Device control 2
147	19	DC3	**CONTROL S**	Device control 3 (X-OFF)
148	20	DC4	**CONTROL T**	Device control 4
149	21	NAK	**CONTROL U**	Negative acknowledge
150	22	SYN	**CONTROL V**	Synchronous idle
151	23	ETB	**CONTROL W**	End of transmission block
152	24	CAN	**CONTROL X**	Cancel
153	25	EM	**CONTROL Y**	End of medium
154	26	SUB	**CONTROL Z**	Substitute
155	27	ESC	**CONTROL .**	Escape
156	28	FS	**CONTROL ;**	File separator
157	29	GS	**CONTROL =**	Group separator
158	30	RS	**CONTROL 8**	Record separator
159	31	US	**CONTROL 9**	Unit separator

You may also obtain detailed CALL KEY subprogram information, including keyboard diagrams, in your *User's Reference Guide* for the TI-99/4A Computer.

CALL VERSION SUBPROGRAM

The VERSION subprogram (discussed in Chapter 4 of your *TI Extended BASIC* manual) now returns a value of 110 on both computers.

DATA STATEMENT

The computer reads any information entered after a DATA statement as a part of the DATA statement. Therefore, in a multi-statement program line, a DATA statement should not be followed by another statement.

SCIENTIFIC NOTATION

Whenever you use scientific (or exponential) notation, be certain that the "E" is an upper-case (large capital) character. A lower-case "e" may cause your program to function improperly.

PRE-SCAN — !@P– and !@P+

After you enter RUN to start a program, you may notice a pause before the program actually begins. This pause is the time the computer takes to "pre-scan" your program to establish memory space for variables, arrays, and data. Then the computer proceeds through each instruction, performs the appropriate functions, and establishes variable values. Since the time required to pre-scan depends on the length of the program, you may want to decrease the pre-scan pause, particularly if you have a long program.

TI Extended BASIC's new pre-scan commands, !@P– and !@P+, allow you to control which instructions will not be pre-scanned. Because the purpose of the pre-scan is to set memory space for variables, only those instructions which contain the first reference to the variables need to be pre-scanned. Therefore, many other instructions in your program do not require a pre-scan.

Careful program planning is required to minimize the statements that need the pre-scan. When certain types of statements (as explained here) are used in your program, the procedures listed below should be included in the pre-scan.

- Enter your *first* DATA statement within the pre-scan.
- Include the first use of each variable and/or array. (Also, include the OPTION BASE statement, if used.)
- Include the first reference to each CALL statement of any subprogram.
- Include all DEF statements for user-defined functions.
- Include all SUB statements and SUBEND statements in the pre-scan.

Note that a variable in a user-defined (SUB) subprogram is considered to be unique from any other variable used elsewhere in your program, even though the name and value may be the same. Therefore, each variable used in a user-defined subprogram must be included in the pre-scan.

To use the pre-scan option, first be certain that your completed program runs successfully. Then, at the beginning of a group of function statements, use the !@P– command to "turn off" the pre-scan. The following statements will not be pre-scanned, allowing the execution of your program to begin more quickly. Any statements related to variable names (not previously referenced during pre-scan) return a syntax error if the pre-scan is "off." Note that !@P– cannot be followed by another statement in a multiple statement.

To resume the pre-scan, simply enter the command !@P+. This command causes the pre-scan to "turn on" and memory space for variables may be set. Remember to use the !@P+ command before a SUB or SUBEND statement and do not incorporate this command as a part of a multiple statement.

You may choose to use the pre-scan feature several times throughout your program. By turning the pre-scan on and off, your program can begin to execute more efficiently. The effectiveness of the pre-scan is more noticeable in large programs than small programs. Note that when using the TI-99/4A Computer, the commands, !@P– and !@P+, may also be entered with a lower-case "p" character.

The following examples illustrate how to include the pre-scan statements in an existing program. The final example demonstrates the most efficient use of the pre-scan feature by making use of a GOTO statement.

Examples:

Original program:

```
100 CALL CLEAR
110 CALL CHAR(96,"FFFFFFFFFFFFFFFF")
120 CALL CHAR(42,"0F0F0F0F0F0F0F0F")
130 .
140 .
150 .
160 CALL HCHAR(12,17,42)
170 CALL VCHAR(14,17,96)
180 DELAY=0
190 FOR DELAY=1 TO 500
200 NEXT DELAY
210 DATA 3
220 .
230 .
```

With pre-scan control added:

```
 10 DATA 3
100 CALL CLEAR
110 CALL CHAR(96,"FFFFFFFFFFFFFFFF")
120 CALL CHAR(42,"0F0F0F0F0F0F0F0F")
125 !@P-
130 .
140 .
150 .
155 !@P+
160 CALL HCHAR(12,17,42)
170 CALL VCHAR(14,17,96)
180 DELAY=0
185 !@P-
190 FOR DELAY=1 TO 500
200 NEXT DELAY
210 .
220 .
230 .
```

Notice that the first DATA statement has been moved to the beginning of the program so that it is included in the pre-scan. By including statements 125, 155, and 185, the pre-scan is turned off and on and off again. This causes the program to begin to execute more quickly.

With GOTO added:

You have the added ability to "trick" the computer into establishing memory space for CALL statements, as well as variable-related statements, without actually performing those statements. To do this, simply use a GOTO instruction in your program. The following example demonstrates the original program adapted with a pre-scan and a GOTO statement.

```
 10 DATA 3
 20 GOTO 100::DELAY::CALL CHAR::CALL CLEAR::CALL HCHAR::CALL
    VCHAR::!@P-
100 CALL CLEAR
110 CALL CHAR(96,"FFFFFFFFFFFFFFFF")
120 CALL CHAR(42,"0F0F0F0F0F0F0F0F")
130 .
140 .
150 .
160 CALL HCHAR(12,17,42)
170 CALL VCHAR(14,17,96)
190 FOR DELAY=1 TO 500
200 NEXT DELAY
210 .
220 .
230 .
```

Note that the GOTO method causes the necessary memory space to be reserved in line 20. However, the statements in line 20 do not execute until they are encountered further on in the program. Thus, as shown in the preceding and following examples, you can put all of your variable references together and your subprogram calls do not have to be syntactically correct. This can be the most efficient use of the pre-scan option.

```
100 GOTO 180::X,Y,ALPHA,BETA,Z=DELTA::DIM B(10,10)
110 CALL KEY::CALL HCHAR::CALL CLEAR::CALL MYSUB
120 DATA 1,3,STRING
130 DEF F(X)=1-X*SIN(X)
140 .
150 .
160 .
170 !@P-
180 .
190 .
200 .
```


PROGRAMMING WITH LOWER-CASE LETTERS

Device names must be entered in upper-case (large capital) letters only. For example, "DSK1" is a correct device name, but "Dsk1" is not. Any reference to a device name spelled in lower-case (small capital) letters results in an error message.

File names are also very specific. Not only are they exact as to the correct spelling, but they are also specific as to the use of upper- or lower-case letters. For example, the file name, MYFILE, is not the same file as Myfile (a combination of large and small capital letters). Any file name listed in part or whole by lower-case letters is not accessible by the TI-99/4 Computer. Only the TI-99/4A Computer can access a program named or called in lower-case letters.

Lower-case letters in DATA statements or quoted strings function correctly and offer a wide variety of programming techniques on the TI-99/4A Computer. However, lower-case quoted strings and data are not displayed if you run the program on a TI-99/4 Computer. If you plan to run your program on both the TI-99/4A and TI-99/4 Computers, take special care when using lower-case letters.

To display the lower-case letters in your TI-99/4A Computer program when the program is run on a TI-99/4 Computer, simply include the following statements. Small capital letters are created similar to those of the TI-99/4A. Be sure to allow adequate memory space and execution time.

```
100 FOR I=65 TO 90
110 CALL CHARPAT(I,A$)
120 B$="0000"&SEG$(A$,1,4)&SEG$(A$,7,4)&SEG$(A$,13,4)
130 CALL CHAR(I+32,B$)
140 NEXT I
```

Insertion of the above program lines into your TI-99/4A program allows pre-programmed lower-case characters to be displayed by the TI-99/4 computer.

SIZE COMMAND

The SIZE example, using the Memory Expansion unit discussed in Chapter 4 of your *TI Extended BASIC* manual, now informs you that you have 24488 "BYTES OF PROGRAM SPACE FREE".

TAIL REMARKS

If you previously programmed a TAIL REMark that is identical to the pre-scan instructions (!@P+ or !@P–), your program will no longer function properly. These groups of characters are now considered to be "reserved words" for the operation of the computer.

CORRECTION TO APPENDIX C

ASCII code 12 in *Appendix C* of your *TI Extended BASIC* manual should be stated as the "PROC'D" character rather than as the "CMD" character.

LARGE PROGRAM FILES

Some programs written with TI BASIC may be too large to run with TI Extended BASIC because TI Extended BASIC requires more system overhead than TI BASIC. If you attempt to load such a program, your system will lock up. Before you can continue, you must turn your computer off, wait several seconds, and then turn it on again.

Entering a CALL FILES(1) or CALL FILES(2) command before loading your program *may* free enough memory to run the program with TI Extended BASIC. (A full explanation of the CALL FILES command can be found in the Disk Memory System manual.)

If a CALL FILES command does not free enough memory, you must shorten your TI BASIC program by deleting statements until the program fits in the memory available with TI Extended BASIC. However, if you have a Memory Expansion unit, you can run the entire program by using the following procedure:

1. As a safety measure, make a backup copy of your TI BASIC program on a cassette tape or diskette.
2. With the Memory Expansion unit attached and turned on, load your program with TI BASIC. Next, delete several statements, and save the shortened program on cassette tape or diskette. Then try to load this shortened program with TI Extended BASIC.
3. Type the deleted statements back into the proper places in your program.
4. Save your program on a diskette only. You are now ready to run your program with TI Extended BASIC and the Memory Expansion unit.

Note: Programs converted in this fashion can only be run with TI Extended BASIC and with the Memory Expansion unit attached and turned on. They are not stored in PROGRAM format.

MEMORY EXPANSION UNIT AND CASSETTE-BASED PROGRAMS

The Memory Expansion unit adds 32K bytes of Random Access Memory (RAM) to the built-in memory of the computer. However, even with the Memory Expansion unit available, the largest TI Extended BASIC program that can be stored on a cassette tape is 12K bytes in size. Note that, although the length of the actual program is limited, utilizing the Memory Expansion unit provides other advantages. For example, with the unit attached and turned on, your program (which can be up to 12K bytes in length) is stored in the expansion RAM. The numeric data generated by the program is stored in the Memory Expansion unit and the string data is stored in the computer's built-in memory. Without the unit, the program must be shorter so that both it and the generated data can be stored in the computer's built-in memory.

CONTINUE COMMAND

A CONTINUE command is used to resume your program when you break by using a BREAK command or by pressing **CLEAR**. However, if your last command (before the CONTINUE command) results in a error, the program may not continue properly. Your final command to the computer before the CONTINUE command must be correct. If you receive an ERROR message, be sure to enter a correct command, such as a PRINT command, before resuming program execution.

MANUAL ERRORS

Page 39

The second sentence in the third paragraph of the "Numeric Constants" section should be corrected to read "...number is greater than 99 *or less than* -99, then ...."

Pages 79 and 150

The string used in a *string-expression* with the DISPLAY ... USING and PRINT ... USING statements may be more general than shown in the examples in the manual. For example, both of the following are valid statements.

```
PRINT USING A$:X,Y
DISPLAY USING RPT$("#",5)&V$:A(12)
```

Pages 89, 133, and 135

The GOSUB, ON GOSUB, and ON GOTO statements should not be used to transfer control to and from subprograms.

Page 114

If you press **CLEAR** when using the LIST command, the listing stops and cannot be restarted.

Pages 118 and 119

The graphic figures at the bottom of page 118 and the top of 119 should be reversed.

Page 185

The TAB function cannot be used in the PRINT ... USING or DISPLAY ... USING statements. Also, the second paragraph of this explanation should be corrected to read as follows: "If the number of characters already printed on the current record is less than *numeric-expression*, the next *print-item* is printed beginning on the position indicated by *numeric-expression*. If the number of characters already printed on the current record is greater than or equal to the position indicated by *numeric-expression*, the next *print-item* is printed on the next record beginning in the position indicated by *numeric-expression*.

Page 200

In Appendix H, Color Combinations, the color codes for the last two listings in the "Best" category should be as follows.

14, *10*	Magenta on Light Red
3, *16*	Medium Green on White

In the "Fourth Best" category, the third combination in the second column should read:

6, 2	Light Blue on Black.

Format Lines

The following list gives corrections that should be made to the indicated formats and also shows the present format information.

DIM Statement (page 76)

Correct Format: (*integer1*[,*integer2*]...[,*integer7*])[,...]
Present Format: (*integer1*[,*integer2*]...[,*integer7*][,...])

DISPLAY Statement (page 77)

Correct Format: [SIZE (*numeric-expression*)]:]*print-list*
Present Format: [SIZE (*numeric-expression*)]:]*variable-list*

DISPLAY ... USING Statement (page 79)

Correct Formats: USING *string-expression*[:*print-list*]
USING *line-number*[:*print-list*]
Present Formats: USING *string-expression*[:*variable-list*]
USING *line-number*[:*variable-list*]

LINPUT Statement (page 113)

Correct Format: #*file-number*[,REC *record-number*]:
Present Format: [[#*file-number*][,REC *record-number*]:]

PRINT ... USING Statement (page 150)

Correct Format: [#*file-number*[,REC *record-number*],]
Present Format: [#*file-number*[,REC *record-number*]]

SPRITE Subprogram (page 173)

Correct Format: *dot-column*[,*row-velocity*,*column-velocity*][,...])
Present Format: *dot-column*,[,*row-velocity*,*column-velocity*][,...])

ADDENDUM TO TI EXTENDED BASIC

SYSTEM LOCK-UP WITH SUBPROGRAMS

A syntax error in a subprogram statement (IE. SPRITE, DELSPRITE, POSITION, COINC, MAGNIFY, MOTION, LOCATE, PATTERN, DISTANCE, SAY, SPGET AND CHARSET) may cause your system to lock-up. Turning the computer console off and on is the only way to restart your system.

INCORRECT ERROR MESSAGE FOR SUBPROGRAMS

The error messages from a CALL Subprogram statement (IE. SPRITE, DELSPRITE, POSITION, COINC, MAGNIFY, MOTION, LOCATE, PATTERN, DISTANCE, SAY, SPGET and CHARSET) are inappropriate. Use the following reference for error messages from the subprograms:

"Syntax Error"	means	"String-number Mismatch"
"Bad Argument"	means	"Bad Value"
"Bad Value"	means	"Incorrect Argument List"

MANUAL ERROR

The program listing on page 153 in the manual is incorrect. Line 110 should read.

```
OPEN #1: "DSK1.RNDFILE",RELATIVE,INTERNAL
```