SECTION 1

ARMADILLO BASIC SPECIFICATION

1.1 General Features

Armadillo BASIC will execute user programs from CPU RAM, with program sizes up to 63K bytes. Each array will be to 64K bytes (8K reals, 32K integers, 16K strings). The stack will be limited to 4K bytes. Otherwise the only memory restriction is the total available RAM.

Extended BASIC will be a true extension of the standard interpreter, rather than a separate interpreter that just happens to share a few routines. Designing it this way should allow a smaller (or more powerful) Extended BASIC, and one that is easier to maintain than the current package.

Execution of BASIC programs from GROM will be supported, but access to subprograms in GPL in command modules will not be supported. PRK, Statistics, S-F Administrative, etc. **WILL**NOT**WORK** because of such statements as CALL A(...).

The following subprograms will be added to console BASIC, to make up for the fact that command module subprograms will not be supported.

* CALL SAY,SPGET (as in the Speech Editor)

* CALL INIT,LOAD,LINK,PEEK,POKEV,PEEKV,CHARPAT (as in the Editor/Assembler)

Notice that BASIC programs that use the PRK subprograms or the Carlisle Phillips "cheater" package will not run.

1.2 Added Features

1.2.1 Screen Access.

A new screen handler will be used. It will support faster scrolling, windowing, and access to all VDP modes. Alternate 80-column video may be provided if a definition of it can be supplied in time.

* CALL SCREEN(background-color) (current usage)

* CALL SCREEN(background, foreground)

* CALL SCREEN(background, foreground, mode)

* CALL SCREEN(background, foreground, mode, left-margin, right-margin, top-margin, bottom-margin)

Modes will include pattern, multicolor, text and bit-map. The default is pattern mode with left and right margins of 2 and top and bottom margins of 0.

CALL LINE(y1,x1,y2,x2,color) will draw a line segment in bitmap or multicolor mode. If the color is omitted, black is assumed.

CALL DOT(y,x,color) will draw a dot of the given color in bitmap or multicolor mode. If the color is omitted, black is assumed.

1.2.2 Sprites.

Sprites will be accessible in all modes except text. They will be accessed as in Extended BASIC, except that 32 sprites will be available.

1.2.3 Integers.

BASIC will implement an integer data type, for speed and space efficiency. The default numeric type is still real.

1.2.4 Multiple-statement Lines.

Standard BASIC will include multiple-statement lines and the corresponding enhancements to the IF-THEN-ELSE statement.

1.2.5 Program Chaining.

Standard BASIC will implement RUN as a statement, for chaining. RUN will be enhanced to accept an arbitrary string expression as an argument, instead of just a constant.

1.2.6 Accept and Display.

Standard BASIC will include a simplified form of accept and display, and a function TERM to return the function code that ended the input (proceed, enter, up, down, etc.). Only the AT and SIZE clauses will be supported.

1.2.7 Loading and Saving Programs.

While BASIC is running, only 24K bytes of RAM can be mapped into the logical address space. Thus memory image saves and loads (i.e. PROGRAM files) will be limited to 24K. (This assumes that all of the DSRs are written so that they can use CPU RAM buffers.) Programs that are larger than 10K will be saved in the same sequential file format that Extended BASIC currently uses for programs too large to fit in VDP RAM. (INTERNAL, VARIABLE 254).
Problem: A cassette cannot currently handle variable files or records longer than 192.
BASIC will also be able to load DISPLAY, VARIABLE 80 text files of the type produced by LIST and by the text editors. This type of loading will crunch each line as it is input. Loading will be slower than the standard binary loads, but this will permit loading foreign programs (from RS232) as well as using text editors on BASIC programs.

1.2.8 Assembly Language Support.

The default condition is that BASIC will use all of available RAM for program and data. Thus the user must allocate some RAM before assembly language can be loaded or accessed. This will be done in the CALL INIT statement.

CALL INIT will allocate 8K bytes of RAM for the user and will load utility routines into the first 2 or 3 K of it. CALL

INIT(X) will allocate X K bytes of RAM for the user. The minimum is 4, and the maximum is 64 (or as much as remains after allowing for the current BASIC program & data and DSR allocations). A value of 3 or less will be treated as CALL INIT(0), which will return all allocated RAM to BASIC.

The utilities will include NUMREF, NUMASG, STRREF, STRASG, LOADER, DSRLNK, GPLLNK, XMLLNK, etc. as in the Editor/Assembler and Extended BASIC. Upon a CALL LINK, the BLWP vector table will be mapped in at >2000 (compatible with Extended BASIC) and all other allocated RAM at >A000. Note that the loader can directly load only 24K of RAM, even though 64K may be allocated. Absolute origin code which loads on the 4A may not load properly on the Armadillo. (e.g. the SAVE utility in the Editor/Assembler or the text-to-speech tables in the disk text-to-speech package.

1.3 Speed Enhancements

The following functions will be faster than 99/4A BASIC, due primarily to translation of GPL to machine code, and extra scratchpad RAM.

* Screen access and scrolling
* Formatting numbers for print (convert-number-to-string)
* Listing and editing programs
* Intrinsic functions (SIN, EXP, etc.)
* String handling
* Random number generation

1.4 Extended BASIC Features

Extended BASIC will still have the following features not found in Standard BASIC:

* User-written subprograms with parameters and local variables

* Program merging

* Functions PI, MAX, MIN, RPT

* Logical operators AND, OR, XOR, NOT

* Block statements DO, LOOP, UNTIL, WHILE, EXIT (From new ANSI standard; new for Armadillo version)

* Full ACCEPT and DISPLAY statements

* Formatted output via PRINT USING and DISPLAY USING

* Tail remarks

* Error, warning, break handling under user control

DIFFERENCES BETWEEN TI EXTENDED BASIC AND ARMADILLO BASIC

```
 DATE     CHANGE
 ====     ======

3/22/83  Numeric variables can be of two types:  REAL or INTEGER.
         All numeric variables are automatically assigned the default
         type of REAL (floating point) unless explicitly declared
         otherwise in a type statement.
         The ALL clause in a numeric type statement changes the default
         type to the declared type.
         Example:
         INTEGER ALL
         The INTEGER ALL statement changes the default type to INTEGER.
         NOTE:  If the ALL specification is used, the statement should
         precede the first variable reference (this includes the DIM
         statement).

3/22/83  REAL (floating point) variables are the same as the Numeric
         Constants in Extended BASIC.
         The formats for the REAL statement are:
         REAL ALL
         REAL variable-name   (where the variable-name is any numeric
                               variable name or array declaration or
                               a list of these things)

3/22/83  The INTEGER statement assigns the INTEGER type to a variable
         or list of variables.
         An advantage to INTEGER variables is that they are processed
         faster and take up less space than REAL variables.
         An INTEGER variable may have a value of +32767 through -32768.
         Formats:
         INTEGER ALL
         INTEGER variable-name   (where the variable-name is any numeric
                                  variable name or array declaration or
                                  a list of these things)
         Example:
         INTEGER A,B,C           (variables A, B, and C are declared to
                                  be type INTEGER)

3/22/83  There are now several ways to dimension an array.
         To dimension an array A to size 10 and type REAL:
         100 DIM A(10)
              or
         100 REAL A(10)
              or
         100 REAL ALL
         110 DIM A(10)
         To dimension an array A to size 20 and type INTEGER:
         100 INTEGER A(20)
              or
         100 INTEGER ALL
         110 DIM A(20)
         If you specify:
         100 DIM A(10)
         110 REAL A(10)
         A name conflict error will result, since you have declared the
         array twice.

3/22/83  User-Defined Functions are 1 program line in length.
         UDFs now support up to seven parameters (instead of one).
         The type of the function can be specified as INTEGER, REAL,
         or $ (Character).
```

UDFs still can not be defined recursively.
Format:

```
DEF function-name[(parameter-list)]=expression
```

Examples:

```
DEF F(X,Y,Z)=expression             (defaults to a REAL function)
DEF INTEGER F(X,Y,Z)=expression     (specifies an INTEGER function)
```

The following is also legal and prints a result of 14.

```
100 DEF F(X,Y)=X+Y+4
110 PRINT F(F(1,2),3)
```

3/22/83 The type of the variables which appear in a parameter-list of a DEF or SUB statement can also be specified as INTEGER, REAL, or $ (Character).
Examples:

```
DEF F(INTEGER X, REAL Y, INTEGER A)=expression
DEF F$(A$,B$,C$)=expression
SUB F(INTEGER A, REAL X)
SUB F(A$,B$)
```

3/22/83 The VALIDATE clause in the ACCEPT statement has the following data-type additions:
ALPHA permits all uppercase and lowercase alphabetic characters.
LALPHA permits all lowercase alphabetic characters.

3/22/83 The VERSION subprogram will return a value of 200.

3/22/83 The FOR-TO-STEP statement executes much faster when a control-variable which has been declared as INTEGER type is used.

3/22/83 When a numerical expression is evaluated, INTEGERS are converted to REAL only when necessary.
This means that when a numerical expression contains only INTEGER values, all evaluation is done in INTEGER arithmetic.
Example:

```
100 INTEGER ALL
    ...
200 C=A+B
```

No conversions are necessary in the above example.
If an expression contains both INTEGER and REAL values, conversion from INTEGER to REAL is done only when necessary.
Example:

```
100 INTEGER A,B,C
    ...
200 C=A+B+3.45
```

In this example, A and B are added, the result is converted to REAL and is then added to 3.45. The final result is then rounded to the nearest integer and assigned to C.

3/22/83 There will be two or three graphics routines which will be added. Further information on these will follow.

3/22/83 The most change will occur in the Graphics Modes.
There will be six modes:

```
Graphics 1 (pattern mode)
Text
Split-1    (Graphics 2 with text on top third of screen)
Split-2    (Graphics 2 with text on bottom third of screen)
Graphics 2 (full-screen)
Multicolor
```

3/22/83 The following subprograms and statements will have some changes due to the way they respond in the various Graphics Modes.
COLOR
HCHAR

VCHAR
ACCEPT
INPUT
DISPLAY
PRINT
GCHAR
The changes will come later.

DIFFERENCES BETWEEN TI EXTENDED BASIC AND ARMADILLO BASIC

```
DATE     CHANGE
====     ======
```

4/26/83 The following describes the various Armadillo screens.

The first screen is the color bar screen which is layed out as follows:

```
+-----------------------------------------+
|                color bar                |
+-----------------------------------------+
                    TI
                  emblem

             TEXAS INSTRUMENTS

           ADVANCED HOME COMPUTER




       READY-PRESS ANY KEY TO BEGIN
+-----------------------------------------+
|                color bar                |
+-----------------------------------------+
      @1983  TEXAS INSTRUMENTS  V1.0
```

The ADVANCED HOME COMPUTER line is subject to change.

The next screen is the menu. This screen currently has three options which are lettered rather than numbered.
The options are:

[A] FOR P-SYSTEM
[B] FOR SET SPEED
[C] FOR TI BASIC

The wording of choice [A] is subject to change.

If [B] is selected, another menu will appear. It also has three options which are also lettered.
The options are:

[A] SLOW
[B] 99/4A SPEED FOR GAMES
[C] FULL SPEED

Option [A] SLOW:
The main advantage to having a slower speed, is that some games execute properly at this speed and therefore could be played at this easier level. However, this is game dependent and varies drastically from game to game. Munchman plays well at this speed, but Parsec does not.

Option [B] 99/4A SPEED FOR GAMES:
Games run at this speed should be the same as they were on the 99/4A.

Option [C] FULL SPEED:
BASIC always uses this speed.
Some games can be played at this speed, but once again it is game dependent.

After the speed selection has been made, the main menu screen reappears.

If the [C] option, FOR TI BASIC, is chosen from the main menu, the following information and prompt appear:

```
TI BASIC Ready V1.0
xxxxxx Bytes free
>
```

The xxxxxx is the number of bytes available, and varies depending upon the amount of memory and the number of devices the system has.
TI BASIC always uses the FULL SPEED speed.

DIFFERENCES BETWEEN TI EXTENDED BASIC AND ARMADILLO BASIC

DATE	CHANGE
4/26/83	It is now possible to change existing line numbers. To change a line number, enter the Edit Mode by typing a line number followed by either FCTN E (UP) or FCTN X (DOWN), and then use the FCTN S (LEFT) and FCTN D (RIGHT) to space over to the digit(s) of line number that you wish to change. Changing a line number does not delete the original line, and so can be used to duplicate a line.
4/26/83	There are now six Graphics Modes. Format: CALL GRAPHICS(mode) Valid modes are: (see below)

4/26/83 There are now six Graphics Modes.
Format: CALL GRAPHICS(mode)
Valid modes are:

Mode	Description
1	Graphics I - 32 col x 24 row grid of pattern positions (8 x 8 pixel blocks). Color (per 8 characters) and a maximum of 32 sprites is available. Alphanumeric and special characters can be used.
2	Text - 40 col x 24 row alphanumeric grid (6 x 8 pixel blocks). Only 2 colors (foreground and background) at a time may be used. No sprites are available.
3	Split I - Split screen where the top 1/3 of the screen displays text (32 x 8 alphanumeric) and the bottom 2/3's of the screen displays high resolution graphics (256 x 128). Color is available for each character in the Graphics I portion of the screen and for each 8 dots on Graphics II portion of the screen. A maximum of 32 sprites is available.
4	Split II - Same as Split I mode, except the top 2/3's of the screen displays high resolution graphics (256 x 128) and the bottom 1/3 of the screen displays text (32 x 8 alphanumeric).
5	Graphics II - High resolution color graphics (256 x 192). The result from the DISPLAY, PRINT, INPUT, ACCEPT, and LINPUT statements is not visible in this mode. A maximum of 32 sprites is available.
6	Graphics III - Low resolution color graphics 64 x 48 blocks (4 x 4 pixel blocks). Each block is individually assigned a color. The result from the DISPLAY, PRINT, INPUT, ACCEPT, and LINPUT statements is not visible. A maximum of 32 sprites is available.

4/26/83 There is a new subprogram, CALL MARGINS, which defines the screen margins.
Format: CALL MARGINS(left,right,top,bottom)
where left,right,top, and bottom are integers which represent the number of columns or rows to indent the screen margins.
This means all 32 columns and 24 rows can be used, or smaller "windows" can be defined as desired.
There can only be one "window" defined at a time.
TI Extended BASIC has only one window which can be defined as:
CALL MARGINS(2,2,0,0)
in Armadillo BASIC.
This subprogram affects the following:
- Scrolling
- CLEAR subprogram
- DISPLAY statement (The position (1,1) is the upper left-hand corner of the current window.)

ACCEPT statement (The position (1,1) is the upper left-hand corner of the current window.)

This subprogram does not affect:

HCHAR subprogram
VCHAR subprogram
GCHAR subprogram
Graphics portions of the screen in modes 3, 4, and 5.
Mode 6 is not affected at all.

4/26/83 There are four graphics subprograms which can only be used in modes 3, 4, and 5.

Format: CALL DCOLOR(foreground-color,background-color)
This specifies the color to use in the DRAW, FILL, HCHAR, and VCHAR subprograms. The default color is black on transparent.

Format: CALL DRAW(type,y1,x1,y2,x2[,y3,x3,y4,x4][,...])
where type is:

```
      1 - For DRAW (using DCOLOR)
      0 - For erase
     -1 - For reverse (no color change)
```

where y is dot-row and x is dot-column.
This routine draws a line from the starting point to the ending point. The line type is specified in the CALL DRAW statement.

Format: CALL DRAWTO(type,y1,x1[,y2,x2][,...])
This routine draws a line from the current position to the point specified. The line type is specified in the CALL DRAWTO statement.

Format: CALL FILL(dot-row,dot-column)
This routine fills all pixels surrounding and including the pixel defined by the dot-row and dot-column given in the CALL FILL statement. This routine fills until a pixel of the foreground color or the screen edge is encountered. Filling may also stop if the figure is too complicated.

4/26/83 Format for the CALL COLOR statement depends on the graphics mode. The format:

CALL COLOR(#sprite-number,foreground[,...])

is valid for all modes, except mode 2.
Additional formats are:

Mode	Format
1	CALL COLOR(character-set,foreground,background[,...])
2	CALL COLOR results in an error.
3	CALL COLOR(character-code,foreground,background[,...])
4	CALL COLOR(character-code,foreground,background[,...])
5	CALL COLOR results in an error, unless for a sprite.
6	CALL COLOR(row,column,color[,...]) where row is 1 to 48 and column is 1 to 64.

The Color Codes have remained the same as in TI Extended BASIC.
The Character Sets for mode 1 have changed.
The new Character Sets are:
Set Number   Character Codes

29	0-7
30	8-15
31	16-23
0	24-31
1	32-39
2	40-47
3	48-55
4	56-63
5	64-71
6	72-79
7	80-87
8	88-95
9	96-103
10	104-111
11	112-119
12	120-127
13	128-135
14	136-143
15	144-151
16	152-159
17	160-167
18	168-175
19	176-183
20	184-191
21	192-199
22	200-207
23	208-215
24	216-223
25	224-231
26	232-239
27	240-247
28	248-255

4/26/83 The CALL SCREEN subprogram has been modified.
Format: CALL SCREEN(background-color[,foreground-color])
This works in all six modes, but the foreground color is only effective in mode 2 (Text Mode).
If the foreground color is not specified, the foreground color is not affected.

4/26/83 The CALL HCHAR and CALL VCHAR subprograms have the same format as in TI Extended BASIC, but have different results depending on the current mode.

Mode	Rows	Columns	
1	1:24	1:32	
2	1:24	1:40	
3	1:16	1:32	Character position on graphics part of screen.
4	1:16	1:32	Character position on graphics part of screen.
5	1:24	1:32	
6	Ignored.		

DATE ==== CHANGE ======

6/28/83 The key codes have changed. These changes are attached. The keyboard on the Armadillo has been modified, again. The QUIT,+,= key has switched places with the _,- key. This is the new keyboard:

```
                                              ------------
                                              | FUNCTION |
                                              | SHIFTED  |
                                              | NORMAL   |
                                              ------------

-------------------------------------------------------------------------
|     | DEL | INS |ERASE|CLEAR|BEGIN|PROCD| AID |REDO |BACK |     |QUIT |
|CAPS |  !  |  @  |  #  |  $  |  %  |  ^  |  &  |  *  |  (  |  )  |  +  |
|     |  1  |  2  |  3  |  4  |  5  |  6  |  7  |  8  |  9  |  0  |  =  |
-------------------------------------------------------------------------
|        |     |     | UP  |     |     |     |     |     |     |     |
|  FCTN  |  Q  |  W  |  E  |  R  |  T  |  Y  |  U  |  I  |  O  |  P  |
|        |  q  |  w  |  e  |  r  |  t  |  y  |  u  |  i  |  o  |  p  |
-------------------------------------------------------------------------
      |     |     |LEFT |RIGHT|     |     |     |     |     |     |     |
      |CTRL |  A  |  S  |  D  |  F  |  G  |  H  |  J  |  K  |  L  |  :  |
      |     |  a  |  s  |  d  |  f  |  g  |  h  |  j  |  k  |  l  |  ;  |
      -------------------------------------------------------------------
      |        |     |DOWN |     |     |     |     |     |     |     |
      | SHIFT  |  Z  |  X  |  C  |  V  |  B  |  N  |  M  |  <  |  >  |
      |        |  z  |  x  |  c  |  v  |  b  |  n  |  m  |  ,  |  .  |
      ----------------------------------------------------------------
                      |                                            |
                      |                    SPACE                   |
                      ----------------------------------------------
```

```
   -----------------
   ¦     ¦         ¦
   ¦  _  ¦    |    ¦
   ¦  -  ¦    \    ¦
-------------------
¦     ¦     ¦     ¦
¦  {  ¦  }  ¦  ~  ¦
¦  [  ¦  ]  ¦  `  ¦
-------------------
   ¦     ¦       ¦
   ¦  "  ¦ENTER  ¦
   ¦  '  ¦       ¦
-----------------
¦     ¦       ¦
¦  ?  ¦ SHIFT ¦
¦  /  ¦       ¦
---------------
```

TI - 21234

DIFFERENCES BETWEEN 99/8 BASIC AND 99/4A EXTENDED BASIC

(1) Integer data type added. 'INTEGER' , 'REAL' keywords added.
INTEGER ALL and REAL ALL statements added.

(2) User defined functions can have seven parameters,
instead of just one.

(3) New function FREESPACE(0). (garbage collection is side effect.)

(4) New function TERMCHAR. Returns code of terminator of last
keyboard readln. (e.g. 13 for enter.)

(5) New function VALHEX, takes string argument, returns integer
after hex to integer conversion.

(6) New subprogram GRAPHICS to change graphics mode

```
    CALL GRAPHICS(1) for pattern mode
    CALL GRAPHICS(2) for text mode
    CALL GRAPHICS(3) for split mode with text at top third
                     and high-res at bottom two-thirds of screen
    CALL GRAPHICS(4) for split mode with text at bottom third
                     and high-res at top two-thirds of screen
    CALL GRAPHICS(5) for hiogh-res mode (full screen)
    CALL GRAPHICS(6) for multicolor mode
```

(7) New subprogram MARGINS to set up a window within text portion
of screen. This window affects PRINT, INPUT, LINPUT, ACCEPT, DISPLAY,
TRACE output, and editing.
It as no effect on HCHAR, VCHAR, CLEAR, GCHAR.
The upper left-hand corner of the window is (1,1) for accept and
display at.

(8) New subprograms DRAW and DRAWTO to draw or erase line segment
on hi-res portion of screen

(9) New subprogram FILL to fill closed figure on hi-res portion of
of screen

(10) HCHAR and VCHAR subrograms enhanced to draw characters on high-res
portions of screens in modes 3-5.

(11) New subprogram DCOLOR to specify colors to use in DRAW, DRAWTO,
FILL, HCHAR, VCHAR

(12) SCREEN subprogram enhanced to support optional second parameter
as foreground color in text mode.

(13) Full 32 sprites available in all sprite subprograms

(14) Full 256 characters available in all ascii subprograms
(CHAR, CHARPAT, HCHAR, VCHAR, SPRITE, PATTERN, etc.)

(15) Full 32 color sets available on pattern mode. Numbering is
somewhat contrived to make it an extension of 4A numbering.

(16) COLOR subprogram has special definition while in multicolor
mode. Parameters are row, column, color. Assigns colors to individual
character code in modes 3 and 4.

(17) GCHAR subprogram has definition that depends on mode.

```
    1,2: ascii character at(y,x) as on 4A
    3-5: 0 or 1, the dot at (y,x) in hi-res portion of screen
    6  : color code of block at(y,x)
```

(18) Screen scrolling only affects current window. In default mode (pattern, margins(2,2,0,0)) no scrolling occurs along screen edges.

(19) CALL CLEAR erases entire screen. DISPLAY ERASE ALL only clears the current window.

(20) INIT subprogram has optional parameter to specify number of bytes to be reserved for assembly language code. Default is 8096. (It was about 6000 in 4A Extended Basic, 32K in console basic with editor/assembler.) Can allocate up to 24,336 bytes.

(21) Execution (by downloading) of grom basic programs is supported, but current grom basic programs (PRK, STATISTICS, etc.) are not.

(22) Basic cannot access GPL subprograms in a software cartridge.

(23) Strings can be 4090 bytes long, instead of being limited to 255.

(24) File records can be up to 4090 bytes long, if the device supports it.

(25) The OLD command will load text files (DISPLAY, VARIABLE 80) in addition to PROGRAM and INTERNAL-VARIABLE 254 memory image files.

(26) protected programs (a) start running as soon as they are loaded (b) erase themselves from memory when they stop running, and (c) are encrypted when they are saved.

(27) The RUN statement can have an arbitrary string expression as a parameter, instead of just a constant string.

(28) Validate types LALPHA and ALPHA added to ACCEPT statement.

(29) Keyboard click is available, with CALL LOAD(VALHEX("84BD"),1). Can be disabled with CALL LOAD(VALHEX("84BD"),0).

(30) CALL LOAD and CALL PEEK cannot be used to access the basic program. (There is a memory mapper, and the program is not entirely mapped in.)

(31) SIZE command is deleted (see FREESPACE)

(32) Bytes put on screen with POKEV will not scroll properly.

(33) Subprograms POKEV, PEEKV added as in editor/assembler.

(34) Linking loader handles REF's, compressed code as in editor/assembler.

(35) More special function keys will terminate an input (BEGIN, BACK, AID, PROCEED, REDO, CLEAR)

(36) Control keys are ignored in an input.

(37) Line numbers may be edited.

LAST UPDATE: 9/07/83

This is a list of programs that will not work on a T.I. 99/8.

Scott Foresman Addition Subtraction II. (To be re-released.)
Disk-based Text to Speech.
TE-II Text to Speech.(TE-III is on the way)
All hybrid programs.
  Personal Record Keeping
  Statistics
  Security Analysis
  Perhaps some Scott Foresman administrative internal stuff.
Speak & Spell and any other programs that use the Speech Editor may not wo
Extended Basic
Anything that uses PRK
  Invoice Management
  Personal Report Generator will work, but doesn't make sense any more.
Disk based Munchan, Tombstone, etc. won't load until we put out a new load

TI - 21234

APPENDIX: Differences between previous BASICs and Extended BASIC II

Since TI Extended BASIC II executes more rapidly than previous BASICs, delay loops do not delay a program as long as expected. Similarly, if a negative duration is used with the SOUND subprogram, the sounds are executed approximately three times as fast as expected.

The INIT subprogram with no parameter allocates 8K bytes of memory, if a parameter is specified, up to 24336 bytes (approx. 24K) may be allocated.
TI Extended BASIC allocated approximately 6K bytes of memory.
The Editor/Assembler with Console BASIC allocated 32K bytes of memory.

The HCHAR and VCHAR subprograms will print outside the margins; but will not scroll. This could cause unexpected results when first entering TI Extended BASIC II since characters displayed with the HCHAR or VCHAR subprograms could appear between the screen edge and the margins (columns 1, 2, 31, and 32). These characters will not scroll, but may be erased with the CLEAR subprogram.

The CHARSET subprogram restores more character codes to their default values than previous BASICs. This includes the lower-case character set.

If a PROTECTed program stops executing, it is now necessary to reload the program in order to run it again.

A CALL to routines in a plug-in module will not work. Also, OPEN #1:"SPEECH",OUTPUT for the Terminal Emulator II will not work.

The INTEGER variable and function type may cause some unexpected results. INTEGERs are written to an INTERNAL file as a 2-byte field; REALs are written as an 8-byte field. This means that INTEGER data type values in a TI Extended BASIC II data file will have to be read as string values with a 99/4 BASIC. Also, INTEGER type functions, return an INTEGER value rather than a floating point value. This would be important if numeric expressions were using INTEGER functions, such as: A = LEN(B$) / LEN(C$), since this will do integer division, rather than floating point division.

Assembly language programs which use absolute addressing with BASIC variables or with program loading, will probably not execute. Relocatable assembly language programs will execute correctly. However, NUMREF may return an INTEGER value rather than a floating point. This is denoted by a >6B in FAC+2. In addition, a string reference error may occur with STRREF since it only allows strings up to 255 characters in length, while BASIC allows strings up to 4090 characters in length.

Also, some assembly language programs worked with errors caused by referring to addresses such as >80xx and >82xx instead of >83xx. These programs will no longer work, since these are now valid addresses in BASIC.

All POKEs and PEEKs as used in previous BASICs will fail. In addition, the ability to PEEK and/or LOAD values into program memory is not predictable. This means LOAD cannot be used to modify a program in memory.

The Value Stack, which contains information like: GOSUB entries, FOR-NEXT entries, and temporary values; is now limited to 3840 bytes (3.75K bytes) rather than 12K bytes less BASIC usage. This means that an infinite GOSUB loop will no longer be able to determine the amount of program space available.

The list of reserved words has been expanded. The following words are additions to the Console BASIC reserved word list. Words preceeded by an asterisk (*) are additions to the Extended BASIC reserved word list. Reserved words may not be used as variable names, and thus may cause an error in existing programs.

```
 ACCEPT
 ALL
*ALPHA
 AND
 AT
 BEEP
 DIGIT
 ERASE
 ERROR
*FREESPACE
 IMAGE
*INTEGER
*LALPHA
 LINPUT
 MAX
 MERGE
 MIN
 NOT
 NUMERIC
 OR
 PI
*REAL
 RPT$
 SIZE
 SUBEND
 SUBEXIT
*TERMCHAR
 UALPHA
 USING
*VALHEX
 VALIDATE
 WARNING
 XOR
```

ADDITIONAL ADDITIONS 9/8/83

The maximum program size is 63,230 bytes. This only includes program text, not any variables, strings, or array space.

The maximum string length is 4090 bytes.

A string array cannot have more than 16383 elements.
An INTEGER array cannot have more than 32767 elements.
A REAL array cannnot have more than 8191 elements.

The maximum program line length is 255 characters, including the line number and spaces. However, after a program line is "crunched" it has a maximum length of 160 characters. (Refer to LINES in TI EXTENDED BASIC II CONVENTIONS for a description of "crunched" program lines.)

Variable names are a maximum of 15 characters in length.

CHANGES TO TI EXTENDED BASIC II

DATE	CHANGE
8/10/83	CHARSET: Only restores character codes 32-95, inclusive. CHARSET restores the default colors to all 256 characters.
8/10/83	GRAPHICS: The default for margins for all modes is 2,2,0,0.
8/10/83	NEW, END, GRAPHICS, BREAK: All 256 characters are restored to their original definition.
8/10/83	DELSPRITE: If the ALL option is specified, all sprites are deleted and can only reappear if they are defined by the SPRITE subprogram. If a sprite-number is specified, the sprite can reappear with the LOCATE subprogram, or be redefined by the SPRITE subprogram.
8/12/83	All warning messages are printed in upper case. (Error messages are still printed in lower case.)
8/12/83	Error messages in High-Resolution and Multicolor modes are printed in Pattern mode.