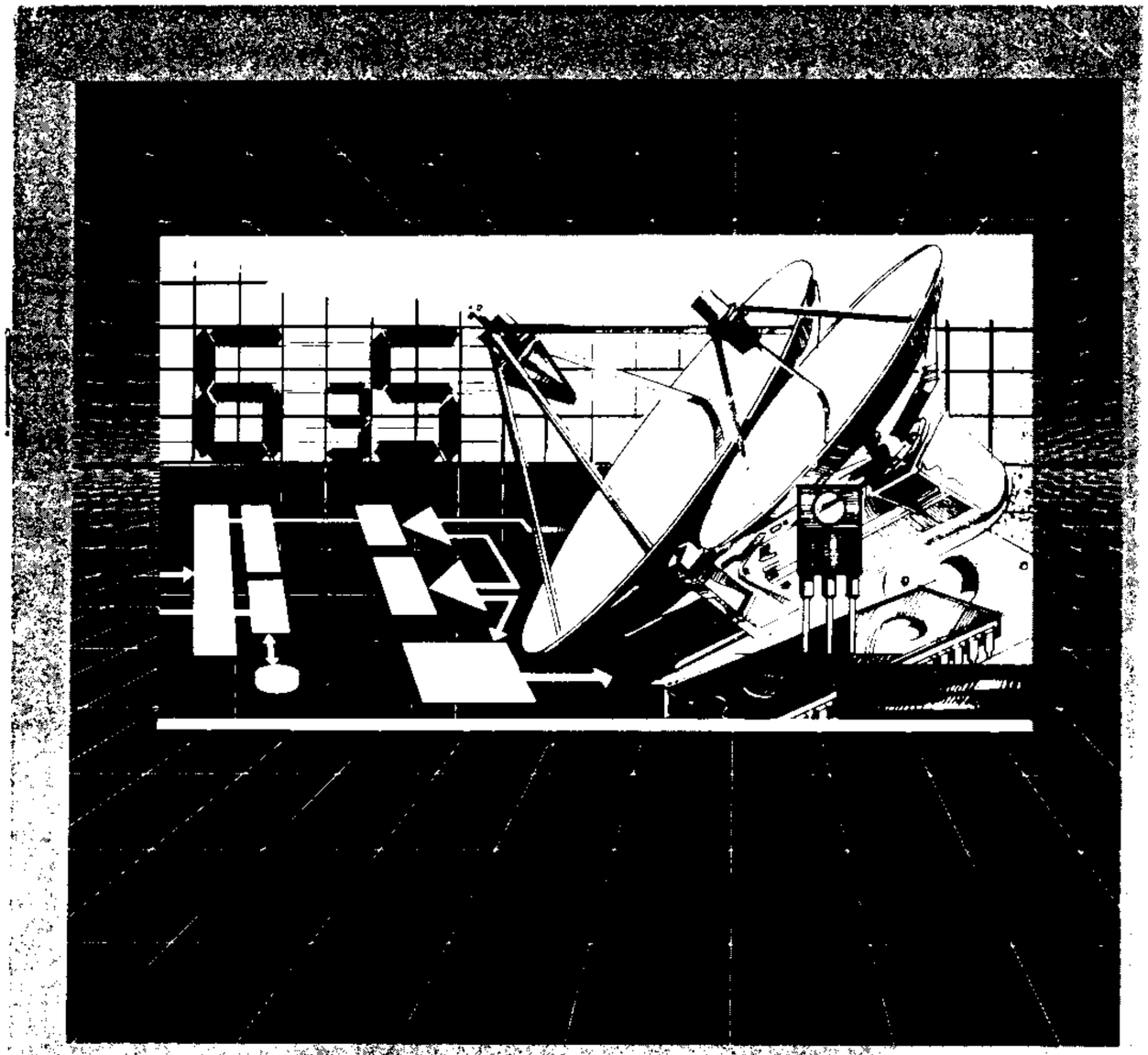
THREE MONTH LIMITED WARRANTY

This limited warranty covers the case components of the software, including cassette, diskette, plastics, and containers. This limited warranty does not apply to the programs contained in the software media and in the accompanying book materials.

Programs are not warrantied to be free from error or to meet the specific requirements of the consumer. The consumer assumes complete responsibility for any decisions made or actions taken based on information obtained using the programs. Any statements made concerning the utility of the programs are not to be construed as expressed or implied warranties. Programs are sold strictly on on "as is" basis and are not warrantied to fulfill a certain purpose or requirement.

In no event shall AMA Software be liable to anyone for special, collateral, incidental, or consequential damages in connection with or arising out of the purchase or use of this product. The sole responsibility of AMA Software, regardless of the form of action, shall not exceed the purchase price of the product. Moreover, AMA Software shall not be liable for any claim of any kind whatsoever by any other party against the user of the products.

Some states do not allow the exclusion or limitation of implied warranties or consequentiial damages; therefore, the above limitations or exclusions may not apply in those states.



AMA-LINK
COMMUNICATIONS PACKAGE

A TERMINAL EMULATOR + UTILITIES

AMA
SOFTWARE

INTRODUCTION

Computers are more and more becoming a part of our lives. One exciting aspect is Tele-Communications, the capability of one computer to talk to another over transmission media (usually the telephone lines).

Four elements are necessary to communicate with other computers: computer, modem, a telephone, and finally a communication program which contains the instructions to allow your computer to talk to the computer on the other end of the phone line.

AMA-Link is a new program that allows your TI-99/4A to act as a remote terminal, thus giving your computer the capability to communicate over the phone lines. As information is sent over the phone lines, AMA-Link picks up the characters and displays them on the screen. AMA-Link also allows you to re-direct these characters to a printer or disk, and even save these characters in a memory text buffer for later storage.

You may use AMA-Link to exchange text files between two computers, access a computer data base, shop by phone, or log onto a local bulletin board system.

SYSTEM REQUIREMENTS

- TI-99/4A computer
- 32K Memory Expansion
- RS232 Interface
- Phone Modem
- Monitor or Television
- One of the following command modules:
  - TI Extended BASIC
  - Editor/Assembler
  - Mini-Memory

A WORD ABOUT COMMUNICATIONS

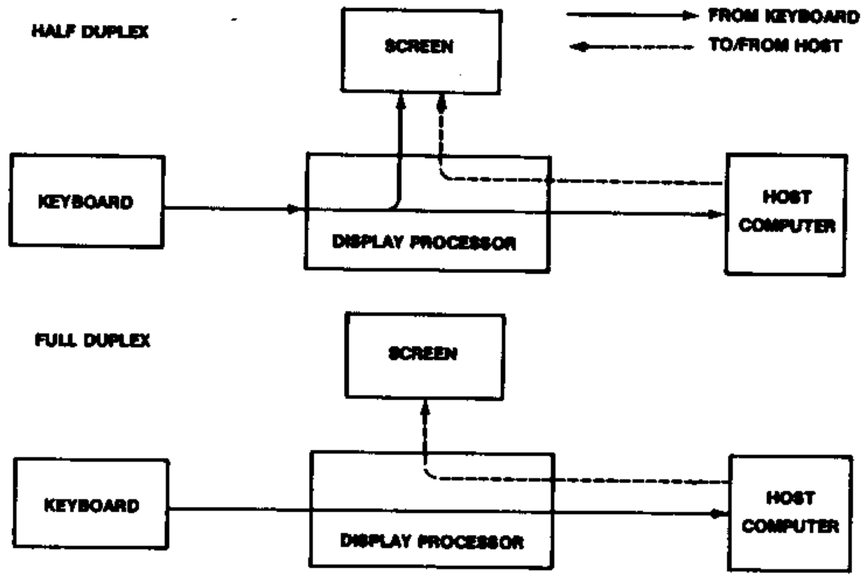
Communications software seems to be misconceived as a very complex, technical programming problem, a concept not especially exact. This section describes the basics of tele-communications.

Tele-communications is simply the passing of characters (letters or numbers) from one computer to another. The character, usually a number or a part of a word, is sent from the originating computer to an RS232 Serial Interface which converts a character consisting of 8 parallel bits to a stream of 8 serial bits, one sent after the other at a certain pre-determined speed.

Converted into a stream of bits, the information is then passed along to a Modem (MOdulator-DEModulator), which converts the bits into sounds. From the Modem, the sound is sent over the phone lines to the receiving Modem, which reverses the process and sends the character to the receiving computer via another RS232 Serial Interface. Upon receiving the character, the receiving computer simply displays the character on the video screen.

Working in this manner, the two computers can converse, or communicate.

The computer usually gets the character to send from the keyboard or from a file located on a disk. If the sending computer is in Half-Duplex mode, it displays the character on the screen before sending it over the phone lines. When in Full Duplex, the computer does not display characters it sends but does display characters it receives from the other computer. This process is called Echo, and most Bulletin Boards and Data Base Services operate with echo. CAUTION: Using Half Duplex when communicating with these host computers causes your computer to display twice each character you type, the echo and the character displayed again by Half-Duplex mode.



GETTING STARTED

1. Make sure your computer is correctly installed and operating.
2. Turn on all the peripherals and then the computer.
3. Insert the floppy diskette in DSK 1.
4. Insert the correct command module.
5. Select the appropriate option. If you are using Extended BASIC, select 1. Terminal Emulator, from the menu that is auto-loaded. If you are using the Editor/Assembler or Mini-Memory module, select the Load and Run option and enter the file name:

DSK1.LOADER

which loads the Terminal Emulator program.

When successfully executed, AMA-Link displays the protocol screen, which allows you to select the protocol you desire or need to communicate with another computer:

1. RS232 Port Number. Enter the port number to which your modem is connected. Valid options are 1 to 4.

2. Baud Rate. The baud rate may be from 110 to 4800 baud. The most popular baud rate, or bit-per-second, is 300 baud. Your TI-99/4A computer is fast enough to accept characters at 300 baud without any trouble. To accommodate speeds greater than 300 baud, the computer uses an interrupt-driven buffer of 256 characters. When this buffer contains more than 200 characters, AMA-Link transmits an ASCII X-OFF character to the sending computer to signal it to pause. After all of the 200 characters have been displayed, AMA-Link sends an X-ON character to the sending computer to signal it to start sending characters again.

When characters are being transmitted, the two computers follow certain conventions to preclude transmission error. These conventions include the number of data bits sent: 7 for straight ASCII or displayable characters only, and 8 for the full range of 8-bit characters (0-255).

First, a Start Bit is sent, then the 7 or 8 Data Bits, followed by a Parity Bit, which helps to confirm that the character has been correctly sent. Finally, 1 or 2 Stop Bits are sent to signify the end of that character. The Echo Mode of Full Duplex causes the characters you send to be sent back (echoed) to your screen by the receiving computer.

Specifying the protocol is the trickiest part of the communication process. You should find out which protocol your host uses. If you do not have access to that information, here are a few combinations. Try them all if you have trouble; usually one of them will work.

a) 8 Data Bits, 2 Stop Bits, No Parity
b) 8 Data Bits, 1 Stop Bit, No Parity
c) 7 Data Bits, 1 Stop Bit, Even Parity
d) 7 Data Bits, 1 Stop Bit, Odd Parity

If a protocol logs onto the host, but still does not provide error-free transmission, try other protocols until you find the one to log on and also transmit correct information.

3. Screen Width. You may specify the screen width to use. Usually you will use 40 columns. If your video display does not accommodate 40 characters, specify 38 or 36.

4. Finally, you may specify an alternate output, which should be used only with an 80-column display card or a high-speed printer. CAUTION: A disk file will NOT work; and, if the alternate output device is not fast enough, characters may be lost or garbled. (The Gemini printer does not function in parallel interface, but does in serial with a buffer.) If you have trouble using this alternate-output feature, use the Text Buffer option described later.

When the protocol screen is first displayed, it shows the default values. To accept these values, simply press <ENTER>; or press Y and <ENTER> to change the values. You may optionally use the Up and Down Arrow keys to edit the protocol screen. After you have made all the changes, enter N at the Change? (Y/N) prompt. This entry activates the on-line Terminal Emulator mode.

ON-LINE OPERATION

During on-line operation, each character received is displayed on the screen. Each character typed at the keyboard is sent over the phone line. If Half-Duplex mode has been activated, the characters typed at the keyboard are also displayed on the screen.

Some special on-line functions enhance the terminal emulation mode. Following is a description of each and how to use each function. These functions are activated by pressing the Control (CTRL) or Function (FCTN) key and the option number.

Reset Protocol CTRL 0 aborts the terminal emulation mode and returns to the protocol screen.

Duplex Switch CTRL 1 allows you to toggle between Half and Full Duplex operation. If you are using Full Duplex,

pressing CTRL 1 switches to Half Duplex operation, and vice verse.

Alternate Output Switch CTRL 2 toggles the operation of the alternate output.

Memory Text Buffer CTRL 3, 4, & 5 control the Memory Text Buffer. This text buffer allows you to save characters that are displayed on the screen to a 16K-character memory buffer and later save that buffer to a printer or disk file.

CTRL 3 resets the text buffer and starts saving characters in the buffer.

CTRL 4 toggles the saving of text to the buffer but does not reset the buffer.

CTRL 5 allows you to save the characters in the text buffer to a disk file, printer, or any other device that supports a DISPLAY,VARIABLE 80 file format.

Using these three keys you can reset the buffer, save characters in the buffer, start and stop saving characters in the buffer, and finally save these characters to a disk file.

Auto Line Feed CTRL 6 toggles the operation of the Automatic Line Feed function (adds a line feed to every carriage return recieved).

Buffer Full Signal When the text memory buffer is almost full, the AMA-Link beeps three times as a warning. When the warning occurs, you have space for 300 characters left in your memory buffer. At 300 baud, you have approximately 10 seconds to stop the sending computer in order to save the text to a disk file. When the buffer is full, AMA-Link honks and stops saving text to the buffer.

Key Click CTRL 9 toggles the key click off or on.

Quit FCTN = terminates program operation.

File Send FCTN 6 allows you to SEND a DISPLAY,VARIABLE 80 text file. After pressing FCTN 6, enter the file name to send. The file is sent just as if you were to type it.

File Save FCTN 7 prompts for a file name to which to save the incoming characters. This feature allows you to send text directly to a disk file. CAUTION: Just before writing a record to the disk, AMA-Link sends an X-OFF character, then writes to the disk. After the record is written, AMA-Link sends an X-ON character to start the transmission again. This option does not work on a host system that does not support X-ON, X-OFF protocol. You will lose characters when the computer is writing to the disk. Usually the first 5 to 10 characters of a new line are lost.

Cancel File Save FCTN 8 closes the file opened by using the FCTN 7 (File Save) function.

Status Display FCTN 9 displays the current RS232 I/O channel being used and the current duplex operation, and finally displays whether the alternate output is activated.

Print Screen Often it is handy to get a hard copy of the monitor display. You may do so by pressing CTRL P. The contents of the screen are printed to the alternate output file specified in the protocol screen. As the alternate output can be either enabled or disabled, you may scroll the screen to display the contents you want to print, then press CTRL P to print an exact copy of the screen.

COMMON PROBLEMS

PLEASE NOTE: When the following file operations:

Alternate Output
File Save

do not function properly with the computer you are communicating with, use the memory text buffer instead. This feature should work no matter which system you are hooked to.

If garbage is displayed on your screen, or some characters are missing, try different protocols. One of the following problems may exist:

1. The host does not support X-ON, X-OFF protocol. Solution: Do not use the File Save feature nor operate above 300 baud.

NOTE: Since AMA-Link uses an X-ON, X-OFF protocol, the host computer may stop sending characters and not recognize the X-ON character to start sending again. If this difficulty occurs, it appears that everything has come to a dead stop. You may then try to start the host computer by pressing CTRL Q which may cause the host to start sending again.

2. Characters are missing at the start of a new line. Solution: Disable File Save and Alternate Output. If this change does not solve the problem, operate at a slower baud rate.

3. Alternate Output does not function with your printer. Solution: Use the Memory Text Buffer.

READING FILES CREATED BY AMA-LINK

AMA-Link creates files in the format: DISPLAY, VARIABLE 80. These file types may be read with TI-WRITER or Editor/Assembler. The disk provides a program that operates in TI BASIC. The name is DSK1.FILES, which allows you to display or print the files received with AMA-Link.

AMA DISASSEMBLER UTILITY

PROGRAM DESCRIPTION: The AMA Disassembler Utility is a programming tool which provides the following utilities:

1. The ability to display the contents of ROM or RAM on the screen in HEXADECIMAL notation, between any two memory addresses,
2. Disassembly of machine code into executable assembly language source code,
3. Conversion of 4-digit hexadecimal numbers into DECIMAL format, and
4. Conversion of Decimal numbers into 4-digit HEXA-DECIMAL format.

PROGRAM REQUIREMENTS: This utility package requires a TI-99/4A console equipped with TI Extended BASIC. (ASCII printer and 32K Memory Expansion are optional.) As the program is supplied on diskette, the appropriate disk system is also required.

PROGRAM OPERATION: To execute the program, simply insert the diskette in drive number 1, then select the Extended BASIC option; the menu program loads and executes automatically. Then select number 2, the Disassembler option; the computer asks for a printer name. Press <ENTER> if you use no printer.

When the prompt: COMMAND? appears, to see a list of commands, enter HELP and the valid commands are displayed on the screen. Following is a list of valid commands and how each operates:

1. DIS -- Disassemble Memory. This command prompts for starting and ending addresses which include the CPU address range you wish to disassemble. Use the following format for entering the address ranges:

>2000,32494

Please note that the numbers may be entered in either base 10 or base 16. To enter a number in base 16 (HEX notation), you must precede the number with '>' to denote HEX notation. Either number may be in HEX or Decimal, and the only stipulation is that the first number must be less than the second number for proper operation. After entry of the starting and ending addresses, the computer starts printing the assembly language source code in the following format:

```
ADDRESS   DATA      OP-CODES  OPERANDS

21F0      3E00      LWPI      >83E0
21F4      2E43      JMP       >21E0
```

NOTE: The source code produced is in non-relocatable code since it consists of disassembled memory locations.

2. DATA -- Display Data. This command displays the contents of memory in HEX notation on the screen. You must enter the starting and ending addresses for the data to be displayed. These addresses are entered as in the DIS command.

3. DEC -- Convert to Decimal. This command converts a hexadecimal number to DECIMAL notation.

4. HEX -- Convert to Hexadecimal. This command converts from decimal to HEXADECIMAL notation.

> The above two commands prompt for the input data and then display both the input data and the converted data. Remember that a HEXADECIMAL number must be preceded by the sign >.

5. HELP -- The Help command displays all of the valid commands.

6. STOP -- This command stops execution of the program.

7. EP -- This command Enables the Printer.

8. DP -- This command Disables the Printer.

NOTE -- To pause during a DIS or DATA listing, press the space bar, and press it again to continue. To abort either of these two commands, press CTRL A. Anytime an error is encountered, the computer displays the message:

ERROR ENCOUNTERED: PLEASE WAIT

and then re-executes the program.

ASSEMBLY LANGUAGE UTILITIES

Extended BASIC Assembly Language Utilities

Before the Utilities can be accessed by your Extended BASIC programs, you must first load the assembly language routines from the diskette into the computer memory. After the routines are loaded into the computer, you simply access them by using the Extended BASIC command CALL LINK with the appropriate parameters for each of the routines provided. Standard routines include:

```
KEYPAD which replaces the BASIC Accept At Statement
KEY    which replaces the CALL KEY Command
```

Other Extended BASIC enhancement routines provided are:

```
TEXT   initializes the 40-Column Mode.
GRAPH  re-initializes the Standard 28-Column Mode.
ASCSET programs the ASCII Code returned by the function keys.
FCTSET equates an input string to a function key.
DISPLY is used as a DISPLAY AT Statement for the 40-Column
         Mode.
CLS    clears the 40-Column screen.
CCLR   sets Screen Colors to be used when TEXT is executed.
CHGKEY allows you to re-program the Utilities' 24 keys to be
         any ASCII characters you desire.
VDPSET allows direct access to VDP row length and column
         offsets used by the routines.
REGWRT allows direct access to VDP registers through BASIC.
```

SEARCH allows you to search a sorted array or sorted, relative disk file.
INSERT allows you to insert an element into any element in an array.
MERGE allows you to merge a sorted array into a sorted, relative record file.

Don't worry if this list of routines seems like a lot to remember and master. The average user will use only two of the above commands, and mastery of as few as six commands can yield outstanding results. We will explain each of these routines in detail but first let's explain how to load them into the computer so that you may access them.

Loading The Extended BASIC Support Routines

The Extended BASIC support routines are provided in two formats. The first is the standard, tagged object code loaded with the CALL LOAD BASIC command. The other is a binary image format of the routines, that saves disk space and loads up to 25 times as fast. Both formats have advantages and disadvantages.

Tagged Object Code

This is the standard format for assembly language routines and may be loaded into the computer with the Extended BASIC Command:

```
CALL INIT :: CALL LOAD("DSK1.KEYPADOBJ") ::
CALL LOAD("DSK1.ROUTINEOBJ")
```

These three commands will load the assembly language routines for you. Once the routines are loaded, they will remain in the computer's memory until you turn off the computer or exit Extended BASIC. The only disadvantage to this method of loading the routines into memory is the load speed.

Binary Image Format

This format is an alternate means of loading the assembly language routines into memory. This method is up to 25 times faster but requires more programming and also creating a short program to load the routines to execute your program with the RUN "DSK1.PROGRAM" command. Following is the load program you should use.

```
100 CALL INIT::REM INITIALIZE MEMORY FOR ASSEMBLY ROUTINES
110 CALL LOAD("DSK1.LOADOBJ") :: REM LOAD ASSEMBLY LANGUAGE
LOAD PROGRAM
120 N$="DSK1.KOBJ" :: REM THIS IS THE NAME OF THE BINARY
PROGRAM FILE
130 REM  THE FOLLOWING TWO LINES TELL THE LOAD PROGRAM THE
FILE NAME OF THE BINARY PROGRAM FILE
140 CALL LOAD(-24565,LEN(N$))
150 FOR A=1 TO LEN(N$)::CALL LOAD(-24565+A,ASC(SEG$(N$,A,1)))
:: NEXT A
160 CALL LINK("LOAD")::  REM NOW LOAD THE BINARY IMAGE PROGRAM
170 RUN "DSK1.PROGRAM" :: REM REPLACE PROGRAM WITH YOUR
PROGRAM'S NAME
```

This small program will load the binary image file for you. You may want to save this program as "LOAD" on your disk so that it will be executed when Extended BASIC is selected. After loading the assembly language routine with either method, you need not reload the routines unless you turn the computer off, return to the color-bar screen, or execute another CALL INIT or CALL LOAD BASIC command. Refer to your Extended BASIC manual for a description to use the standard CALL INIT :: CALL LOAD statements to load the program the first few times you use it.

Programming

Now that the assembly language routines are loaded into memory, we may explain the programming process. The first two commands are the only two programs required to interface the SpeedKey with your programs. The others are provided simply as enhancements that you may use in your programming.

KEYPAD Statement

The KEYPAD statement replaces every ACCEPT AT or INPUT statement currently used in your programs. As KEYPAD works exactly like the ACCEPT AT Extended BASIC statement, please read about it in your Extended BASIC manual.

The format for KEYPAD is:

```
CALL LINK("KEYPAD",IN$,ROW,COL,SIZ,V$)

where: IN$ is the string to be input
       ROW is the row of input
       COL is the column of input
       SIZ is the size of the input
       V$  is the validation string for the input
```

If you use a negative SIZ amount, the program assumes a default input. V$ should contain all of the characters that may be entered. For example, if you want to input a number, use a V$ of "1234567890-.", which allows only those numeric characters, a hyphen, and a decimal to be entered. If the user presses a wrong key, the computer sounds a low beep, indicating the character is not accepted. If you do not wish to use a validation string, simply use a null string (a null string has no characters in it, ""). Following are some examples showing how to convert an Extended BASIC ACCEPT AT statement into the Utility format:

```
BASIC  -- ACCEPT AT(24,1)BEEP SIZE(12):B$
KEYPAD -- CALL LINK("KEYPAD",B$,24,1,12,"")
```



```
BASIC--ACCEPT AT(18,23) BEEP SIZE(-6) VALIDATE ("1234567890")
:A$
KEYPAD--CALL LINK("KEYPAD",A$,18,23,-6,"1234567890")
```

KEY Statement

This statement replaces the CALL KEY statement in Extended BASIC. The format for the KEY command is:

```
CALL LINK("KEY",ASCII,STAT)

where: ASCII is the ASCII code returned by the
           routine
       STAT is the STATUS variable and is the same
           as that used with CALL KEY
```

Following is an example showing both the Extended BASIC statement and the KEY statement:

```
BASIC --  CALL KEY(0,ASCII,STAT)
KEY   --  CALL LINK("KEY",ASCII,STAT)
```

Enhancement Routines

Following is a short description of each enhancement routine you may use to further enhance your programming:

TEXT Command

The TEXT Command simply formats the screen for 40-column text mode. When you use this mode, you must use the routines provided in this Programming section to process any screen activity. The KEYPAD routine compensates for the 40-column mode so that it works the same as the standard mode. The format for the TEXT statement is:

```
CALL LINK("TEXT")
```

GRAPH Command

The GRAPH Command simply reverses the effect of the TEXT Command and returns control to the standard graphics mode.

```
CALL LINK("GRAPH")
```

ASCSET Command

This command allows you to program the ASCII character code returned by the 5 function keys (F1-F5).

```
CALL LINK("ASCSET",KEY,ASCII)
```

where: KEY is the number (F1-F5) of the key to be programmed
ASCII is the ASCII code to be equated to that function key

FCTSET Command

This command allows you to equate a string of up to 40 characters for each function key.

```
CALL LINK("FCTSET",KEY,S$)
```

where: KEY is the function key number (F1 - F5)
S$ is the string to be equated to the key

DISPLY Command

This command operates like the DISPLAY AT Command in Extended BASIC except that it works for either 28- or 40-column mode.

```
CALL LINK("DISPLY",ROW,COL,S$)
```

where: ROW is the row of display
COL is the column of display
S$ is the string to be displayed

Following is an example showing the conversion from Extended BASIC to Utilities procedure:

```
BASIC  -- DISPLAY AT(23,12,):A$
DISPLY -- CALL LINK("DISPLY",23,12,A$)
```

CLS Command

The CLS command functions the same as the CALL CLEAR BASIC command, except it is used only in the 40-column mode. While using the 40-column display mode, never execute a CALL CLEAR statement, because it will yield unpredictable results. The format for the CLS command is:

```
CALL LINK("CLS")
```

CCLR Command

This command allows you to designate the screen colors used when the TEXT command is invoked.

```
CALL LINK("CCLR",CC)
```

where: CC is the color code, calculated thus:
CC = (foreground color code x 16)
+ (background color code)

CHGKEY Command

You may optionally change the ASCII codes returned by every one of the keys on the Utilities' numeric entry pad.

```
CALL LINK("CHGKEY",A$)
```

where: A$ is a 32-character string containing the ASCII codes you want programmed into the keys

VDPSET Command

This command is rarely used by the BASIC programmer. The VDPSET command allows the user to specify the actual row length and column offset factors used by the program. This routine may be useful for someone using special VDP RAM addresses, who wants to access the Utilities. The format for the command is:

```
CALL LINK("VDPSET",ROWLEN,COLOFF)

where: ROWLEN specifies the row length
       COLOFF specifies the column offset factor
```

REGWRT Command

This command allows you to write directly to a VDP Register to change colors after the TEXT mode has been invoked.

```
CALL LINK("REGWRT",REGDATA)

where: REGDATA is a 16-BIT integer value, with the
       most significant byte designating the
       register to write to and the least
       significant byte containing the information
       to be written
```

CAUTION: Take precautions when using this routine, as it may lock-up or confuse your computer and require you to turn it off and on again, thus losing the volatile memory.

Commands for Experienced Programmers

The following commands allow very powerful data manipulation; however, they are not as easy to use as the previous utilities. They are included for experienced programmers only. A good working knowlede of assembly language interfacing is necessary to gain full use of these routines.

SEARCH Command

This command allows you some very powerful array and disk file searching.

Array Handling Format:

```
CALL LINK("SEARCH",X$( ),M,G,P$,T,L,TY)

where: X$( ) is the array to be searched
       M    is the element of the closest match
       G    is the search flag (if non-zero, a match was
            found)
       P$   is the string to search for
       T    is the number of elements in the array
       L    is the number of characters to compare for
            match (valid for disk files only but must
            always be included in command)
       TY   tells the assembly language which format you
            intend to use and must be Ø for array search
```

Disk Search Format:

```
CALL LINK("SEARCH",FD$,M,G,P$,T,L,TY,FN$)

where: FD$ is the string variable the routine returns with
           the string on the disk file that is the
           closest match
       M   is the record in the disk file that is the
           closest match
       G   is the search flag (if non-zero a match was found)
       P$  is the string to search for
       T   is the number of records in the disk file
       L   is the length of the strings in the file and must
           be 1 less than the file length
       TY  is the type of search and must be 1 for a disk
           file search
       FN$ is the name of the file to be searched and must
           be of the following format:
               RELATIVE, INTERMAL, FIXED L+1
```

The Type Variable (TY) also returns any error code encountered during the search. If you choose to use this routine to search a file, you must observe a few precautions:

1. The file should contain only one string per record.
2. The file should be in sorted order (alpha-numeric).
3. The file must have a length of 1 greater than the length specified in the call. The length variable tells the routine how long the string to be searched is and the extra byte needed is simply the length byte used by Extended BASIC.
4. The file must be opened by the Extended BASIC program prior to calling the search.
5. The most common way of handling these limitations is to concatenate a pointer, onto the end of the sorted string, which points to a record in another file that contains the rest of the information.

INSERT Command

This command allows you to insert an element into an array. This Insert utility, used in conjunction with the Search utility, allows you to keep a fully sorted array in memory, and then merge the array into a sorted disk file with the Merge Command described next. These commands allow you to keep a sorted disk file without ever really having to sort the file. The format for the INSERT command is:

```
CALL LINK("INSERT",I$,P,E,X$( ))

where: I$ is the string to be inserted
       P  points to the element where I$ is to be inserted
       E  is the number of elements in the array
       X$( ) is the array where I$ is to be inserted
```

With this utility, you see it is very easy first to use the SEARCH utility to find the proper element for insertion and then to use the INSERT command to actually insert the element into the array.

MERGE Command

The MERGE Command allows you to merge a sorted array into a sorted disk file.

```
CALL LINK("MERGE",X$( ),FR,FLG,FN$,AE,L,E)

where: X$( ) is the sorted array to be merged
       FR  is the number of records in the file
       FLG is the completion flag (if non-zero, an error has
             occurred and the error number is in E)
       FN$ is the name of the OPENED file which is sorted and
             has a length of L+1
       AE  is the number of elements in the array
       L   is the compare length and is 1 less than the file
             length
       E   is the error code returned when FLG is non-zero
```

Please note that this routine is invalid if the array has only one element or the file contains no records. An error 24 may be returned in these cases.

You should familiarize yourself with the following commands before attempting any major programming. Refer to your Extended BASIC manual:

CALL INIT CALL LOAD CALL PEEK CALL LINK

These procedures will aid you in using the Utility Routines.

Assembly Language Support

The Assembly Language Object Code for the Utilities is included on the Software Diskette. You may incorporate it into your programming. You may use the program as often as you like. If you choose to market programs containing the software routines of the Utilities, contact:

AMA Software
P O Box 3Ø24
Springfield MO 65Ø8

concerning royalty information.

Using The Sample Application Program

Included on your diskette is a sample program called CASH CONTROLLER, which incorporates the Utilities into a simple accounting, forecasting, and costing program. The program loads automatically when Extended BASIC is selected from the main selection screen. The program allows you to enter several different amounts, descriptions, and operators to build a single-column spreadsheet. The program is useful in recording expenses, keeping track of your checkbook, balancing budget, and handling many other calculating tasks. After you enter information, you may change it, print it, and optionally save and reload the information to and from a floppy disk. If you would like a complete manual for the program, send a check or money order for $5.ØØ to cover the cost of shipping and handling to AMA Software at the above address, and we will be glad to send you the complete documentation for the program.

After you have used the program, you should list it and note the use of the Utilities' routines. Please remember that the software contained on the supplied diskette is copyrighted material; any unauthorized duplication is prohibited.