```
                 ***   *   *  ****.
                *   *  *   *   *  *.
                *       * *    *  *.
                 ***     *     ***.
                    *   * *    *  *.
                *   *  *   *   *  *.
                 ***   *   *  ****.
..
                      INTRO.1.0.
..
..
.         DATA BASE.
.
.        SUBROUTINES.
..
                      DB.1.0.
..
..
.       STRING ARRAY.
.
.        SUBROUTINES.
..
                      SA.1.0.
..
..
.          STRING.
.
.        SUBROUTINES.
..
                      SM.1.0.
..
..
.          INTEGER.
.
.        SUBROUTINES.
..
                      IN.1.0.
..
..
.          V.D.P..
.
.        SUBROUTINES.
..
                      VM.1.0.
..
..
.     MISCELLANEOUS.
.
.      SUBROUTINES.
..
                      MISC.1.0.
..
..
.          INDEX.
..
                      INDEX.1.0.
........................
```

{I N T R O D U C T I O N}.
.
SUPER EXTENDED BASIC (SXB) is a powerful extension to the TI Extended BASIC programming language. SXB is comprised of over 100 TMS9900 Assembly Language subroutines which substantially expand your programming capabilities on the TI Home Computer. The subroutines are easily invoked with the Extended BASIC LINK subprogram. In essence, it is now possible to access the raw power of assembly alnguage with only a knowledge of Extended BASIC and the information provided in this instruction book. All SXB subroutines are invoked with the following format:.
.

```
     CALL LINK(subroutine-name[,argument-list]).
```

.
For the sake of simplifying the argument-list names, we have devised an easy method of identifying what is required. The format is as follows:.
.

```
     Prefix:     I     Input (may be an equation).
                 O     Output (may not be an equation).
                 IO    Input + Output (may not be an.
                       equation).
.
     Body:       all lower case descriptor.
.
     Suffix:     $     String.
                 $()   String Array.
                 {   }   Numeric Variable.
                 ()    Numeric Array.
```

.
{EXAMPLES:}.

IOdatabase$() would indicate that a string array will be used for both input and output and that it is a "data base." You would enter a value such as MAILLIST$()..
.

Iarray() would indicate a numeric array to be used only for input..
.

Ihexadecimal$ would indicate a (readable) hexadecimal string to be used for input only..
.

Obinary$ would indicate a binary string to be used for output..
.
Subroutine nemes are a maximum of six characters in length. With the exception of the six miscellaneous subroutines (BANNER, KEY1, PARMSV, PARMGT, QUIT, and USRSUB), the first two characters are a prefix representing the category of SXB subroutine it falls into. The five major SXB subroutine categories are as follows:.

..

```
     {PREFIX}     {CATEGORY}.
     DB         Data Base.
```

SA	String Array.
SM	string.
IN	Integer.
VM	V.D.P. -- Video Display Processor.

.
This instruction book has been designed so that it can be updated as needed. Also you may find it advantageous to rearrange the sections of this book to fit your own preferences..
.

{SECTION}	{CATEGORY}.
1	Introduction.
2	Data Base Subroutines.
3	String Array Subroutines.
4	String Subroutines.
5	Integer Subroutines.
6	V.D.P. Subroutines.
7	Miscellaneous Subroutines.
8	Index.

.
.
.

INTRO.3.0.

..

{HOW TO LOAD THE PROGRAM}.

.
The TI Extended BASIC command module should be inserted into the console. Put the SXB disk in drive one. Select EXTENDED BASIC and the subroutines will be loaded automatically. Optionally the SXB disk may be mounted on drive 2 or 3 in which case you would use the RUN "DSKn.LOAD" instruction where "n" is the drive
..
number. {The SXB subroutines must be loaded prior to} {loading your program which calls them.} You will normally not have to reload SXB between programs..
..
.
.
.

INTRO.4.0.

..

{ADDITIONAL MEMORY REQUIREMENTS}.

.
SXB utilizes the entire 8K portion of the 32K memory expansion. Additionally, it also utilizes the first three 256 byte blocks of memory in the 24K portion of the 32K memory expansion. The first block is reserved space for the addition of user defined subroutines (USRSUB) after SXB is loaded. The next two blocks are buffers for input and/or output of strings. CAUTION: {Some SXB subroutines require additional memory over} {and above the aforementioned 768 bytes. In most cases} {it is only for temporary use -- but with the DBKEYS} {subroutine, it stays in effect until another DBKEYS} {subroutine is called or SXB is reloaded.}.
.
Extended BASIC does not normally allow you to use the 8K portion of the 32K memory expansion, so you have

not lost any memory here. But you do loose some of
the 24K portion as explained above. To make up for
this you will find that SXB subroutines do, in many
instances, allow you to do the same amount of
programming without taking up as much memory.
Additionally, the Integer subroutines allow for the
reduction of space needed to store integers. Overall
you should not notice the reduction in memory from the
24K portion of the memory expansion unless you are
sorting large Data Bases..
.

{COMMAND MODE}.

.
All SXB subroutines are designed so that they can be
called in command mode. Except for the few cases
where the computer resets certain values (e.g.,
colors, sprites, etc.), you now have some very
valuable debugging tools. Probably one of the most
powerful is SAVIEW which allows you to inspect the
contents of a string array. This would be
particularly valuable if you interrupt a program and
then need to check the values in a string array..
.
.

..
1.6. Examples of SXB coding
The following few examples of SXB coding should help give you a better idea of just how simple it is to use
the SXB calls. Refer to the appropriate subsequent sections of this reference manual to see the standard
example abbreviations used for illustration purposes.

```
DBKEYS and DBDEL
CALL LINK("DBKEYS",CHR$(25)&CHR$(1)&CHR$(5),V)
CALL LINK("DBDEL","SC01D",V)
SAICNT (page 31)
CALL LINK("SAICNT",DB$(),CNT)
SMTRIM (page 46)
A8="ABCXYZABCABCABCXYZZZZYXZYYXZ"
CALL LINK("SMTRIM",A$,"XYZ")
INPAK4 (page 51)
A=25
B=-24000
C=768
D=1024
CALL LINK("INPAK4",E,A,B,C,D)
VMWNDW (page 62) and VMLIST (page 64)
CALL LINK("VMWNDW",3,3,22,30)
CALL LINK("VMLIST"," NOW IS THE TIME FOR SXB")
BANNER (page 75)
CALL LINK("BANNER",AS(),"SXB")
KEY1 (page 76)
CALL LINK("KEY1","IRDU",V)
PARMGT (page 77)
CALL LINK("PARMGT",PARM$)
......................
```

{D A T A B A S E}.
{C O N S I D E R A T I O N S}.

.
An SXB Data Base (henceforth referred to as Data Base) is made up of a string array and is considered to include only item one through the first item containing a null string (i.e.,"") in the array. If item one contains a null string, then the Data Base is considered to be "empty" even though other items might contain values other than null strings. If you are using OPTION BASE 1, you cannot access item zero... but if you are using OPTION BASE 0 (the Basic default value), you may use zero for any purpose -- just remember that it will be ignored by SXB subroutines. The following is a sample Data Base which will be used to demonstrate the SXB Data Base subroutines:.
.

{SAMPLE DATA BASE}.

.
```
ITEM   VALUE                          NOTES.
  0    "any value"                    Ignored by SXB.
_______________________________________________________.
  1    "SC01DSUPER CATALOGER"         Only item one.
  2    "SXB1DSUPER EXTENDED BASIC"    through the first.
  3    "VT01TVIDEO TITLES I"          item with a null.
  4    "VT01DVIDEO TITLES I"          string are.
  5    "VT02DVIDEO TITLES II"         considered as the.
  6    "VT03TVIDEO TITLES III"        Data Base by SXB..
  7    "VT03DVIDEO TITLES III".
  8    "".
_______________________________________________________.
  9    "any value"                    Ignored by SXB..
 10    "any value".
etc..
```
.
So as to give some meaning to the SAMPLE DATA BASE, here is a possible description to each item: Positions 1-5 are for product numbers. Position five also indicates "T" for tape versiln or "D" for disk version. Positions 6-25 reflect the product name. Note that this last field is variable in length. All Data Base subroutines treat Data Base items as fixed length. Any item shorter than the maximum length specified by the DBKEYS subroutine will be expanded to the maximum length with null characters (i.e., CHR$(0)) for the purpose of doing comparisons. Any item returned to a Data Base will have all trailing null characters deleted..
.
.
.
.

..

{DATA BASE SUBROUTINES}.

.
```
DBDATA  Provide error information concerning other.
        Data Base subroutines..
DBKEYS  Describe Data Base (maximum item length + key.
        fields)..
DBSORT  Sort Data Base..
DBDEL   Delete all Data Base items with keys equal to.
        argument item..
DBINS   Insert new item in Data Base..
DBREPL  Replace item in Data base..
```

```
DBUPUT  Update (replace/insert) item in Data Base..
DBEQCT  Count items in Data Base with keys equal to.
        argument item..
DBFIND  Find next item in Data Base with keys equal.
        to argument item..
DBCOMP  Compare two Data Base formatted items..
DBOKEY  Build a new Data Base with only key fields as.
        output..
.
The last parameter of each Data Base subroutine is
Ovector.  After executing any of these subroutines,
Ovector will be returned with a value which can be
used with an ON Ovector GOTO or GOSUB.  A value of "1"
returned in this numeric variable will always indicate
that an error condition occurred.  Use the DBDATA
subroutine to determine the specifics of the error..
..
                      DB.3.0.
..
.                     DBDATA.
.
CALL LINK("DBDATA",Osubroutineid,Oerrorcode,Ovalue1,
     Ovalue2).
.
This subroutine returns information about the last SXB
Data Base subroutine called.  If the subroutine was
called without an error, Oerrorcode, Ovalue1 and
Ovalue2 will contain zeroes.  Other values are
explained for each particular Data Base subroutine.
..
                      DB.4.0.
..
.                     DBKEYS.
.
CALL LINK("DBKEYS",Idescription$,Ovector)
.
This subroutine defines items in a  Data Base  and how
..
they are to be compared.  CAUTION: {This subroutine}
{must be invoked properly prior to calling any other}
{Data Base subroutines.} Once set, it remains in
effect until DBKEYS is invoked again or SXB is
reloaded..
..
.
The first position of Idescription$ indicates the
maximum length allowed for each item in the Data Base.
This is the format CHR$(maxlength).  The subsequent
optional pairs of positions within Idescription$
identify the key fields within each item.  Only the
key fields will be used for the purposes of comparison
in other Data Base subroutines.  The format for
specifying each key field is CHR$(firstpostion)&&
CHR$(keylength).  Optionally you can specify that a
key is to be treated in descending rather than
ascending order simply by changing the format to CHR$
(firstposition)&&CHR$(keylength+128).  The maximum key
length is 127 characters.  If no keys are specified,
SXB assumes that there is only one key that starts in
```

```
position one and it has a length equal to maxlength or
127, whichever is lesser.  Following are three
EXAMPLES of Idescription$:.
.
Idescription$ VALUE   MAXIMUM   KEY    KEY  ASCENDING/
                         ITEM FIRST LENGTH  DESCENDING
                       LENGTH   POS         SEQUENCE.
.
CHR$(20)                   20     1     20  ascending
.
CHR$(25)&&CHR$(3)&&CHR$(5)   25     3
    5  ascending .
.
CHR$(50)&&CHR$(1)&&CHR$(9)   50     1
    9  ascending .
&&CHR$(24)&&CHR$(5+128)            24
    5  descending
&&CHR$(11)&&CHR$(3)                11
    3  ascending
.
A maximum of 127 key fields is possible with SXB.  The
use of many keys does not substantilly slow down the
comparison process since most comparisons will result
in an unequal condition within the first few
characters.  Keys may overlap if necessary with the
exception that an ascending key may not overlap a
descending key..
..
                      DB.5.0.
..
DBKEYS (Continued).
.
{ADDITIONAL MEMORY REQUIREMENTS}.
.
An additional amount of the 24K portion of memory is
required to store the DBKEYS information.  The amount
is equal to approximately the length of Idescription$
+ 2 [+maxlength +2 if  there  are  any   descending
key fields].  This additional memory will continue to
be used until it is replaced by a different amount
required for the nest call to DBKEYS.  If DBKEYS is
invoked with an error, this additional memory
requirement will be freed up..
.
Ovector values:  1 = Error..
                 2 = O.K..
.
DBDATA VALUES    DESCRIPTION OF ERROR.
50 2 0 0         Idescription$ length invalid..
50 3 0 0         Maximum item length zero.
50 4 n 0         First position of a key is zero..
50 5 n 0         First position of key is greater than
                 maximum item length..
50 6 n 0         Key length zero..
50 7 n 0         Key extends past end of maximum item..
50 8 c 0         Overlapping ascending and descending.
                 key fields..
.
                 n = position within Idescription$..
                 c = position within item..
..
                      DB.6.0.
..
.                     DBSORT.
.
CALL LINK("DBSORT",IOdatabase$(),Ovector).
.
This subroutine sorts all items in IOdatabase$()
according to the key fields specified with the DBKEYS
subroutine..
.
Ovector values:  1 = Error..
                 2 = O.K., no duplicate keys detected..
                 3 = O.K., duplicate keys detected..
.
DBDATA VALUES    DESCRIPTION OF ERROR.
51 1 0 0         DBKEYS improperly/not invoked..
51 9 i m         IOdatabase$() item has length greater.
                 than maximum item..
.
                 m = maximum item length..
                 i = index to item in Data Base..
.
{ADDITIONAL MEMORY REQUIREMENTS}.
.
An additional amount of the 24K portion of memory is
temporarily required to perform the sort.  The amount
is equal to approximately (number of items in Data
Base + 1) * maximum line length.  If you do not have
enough space to use DBSORT, you can still accomplish
the sort (although it is slower) by using DBUPDT as
you add each item to the Data Base..
.
------------------- E X A M P L E --------------------
.
{INPUT:}.
.
DBKEYS Idescription$ = CHR$(25)&&CHR$(1)&&CHR$(5).
.
IODatabase$() = SAMPLE DATA BASE.
.
{OUTPUT:}.
.
IODatabase$() = ITEM   VALUE.
                  1    "SC01DSUPER CATALOGER".
                  2    "SXB1DSUPER EXTENDED BASIC".
                  3    "VT01DVIDEO TITLES I".
                  4    "VT01TVIDEO TITLES I".
                  5    "VT02DVIDEO TITLES II".
                  6    "VT03DVIDEO TITLES III".
                  7    "VT03TVIDEO TITLES III".
                  8    "".
.
Note that the disk versions of Video Titles I && Video
Titles III are now listed before the tape versions..
.
.
                        DB.7.0.
```

```
..
.                      DBDEL.
.
CALL LINK("DBDEL",IOdatabase$(),Iitem$,Ovector).
.
This subroutine searches all items in IOdatabase$()
and deletes all items which have key fields identical
to those of Iitem$..
.
Ovector values:  1 = Error..
                 2 = IOdatabase() item(s) deleted..
                 3 = No equal items in IOdatabase$()..
.
DBDATA VALUES    DESCRIPTION OF ERROR.
54 1 0 0         DBKEYS improperly/not invoked..
54 3 0 0         Iitem$ length zero..
54 9 0 m         Iitem$ length greater than maximum.
                 item length..
54 9 i m         IOdatabase$() item has length greater.
                 than maximum item length..
.
                 m = maximum item length..
                 i = index to item in Data Base..
.
------------------- E X A M P L E --------------------
.
{INPUT:}.
.
DBKEYS Idescription$ = CHR$(25)&&CHR$(1)&&CHR$(5).
.
IODatabase$() = SAMPLE DATA BASE.
.
Iitem$ = "SC01D"
.
{OUTPUT:}.
.
IODatabase$() = ITEM   VALUE.
                  1    "SXB1DSUPER EXTENDED BASIC".
                  2    "VT01DVIDEO TITLES I".
                  3    "VT01TVIDEO TITLES I".
                  4    "VT02DVIDEO TITLES II".
                  5    "VT03DVIDEO TITLES III".
                  6    "VT03TVIDEO TITLES III".
                  7    "".
.
Note that old item 1 has been deleted and all other
items have moved up one position.  If there had been
more than one item with the same key "SC01D", they
..
would have all been deleted.  CAUTION: {If you had}
{previously located an item with DBFIND, and you want}
{to delete only that one specific item, it would be}
{more appropriate to use SADEL to insure that no
others} {are inadvertently deleted}..
.
.
                        DB.8.0.
..
.                      DBINS.
```

```
.
CALL LINK("DBINS",IOdatabase$(),Iitem$,Ovector).
.
This subroutine searches all items in IOdatabase$()
until it finds the first item with a higher key than
that of Iitem$.  Once found, it inserts Iitem$ into
IOdatabase$() moving all subsequent items down one
position.  If an item with a higher key is not found,
Iitem$ will be placed at the end of IOdatabase$().
This subroutine assumes that IOdatabase$() has
previously been sorted..
.
Ovector values:  1 = Error..
                 2 = O.K..
.
DBDATA VALUES    DESCRIPTION OF ERROR.
55 1 0 0         DBKEYS improperly/not invoked..
55 3 0 0         Iitem$ length zero..
55 9 0 m         Iitem$ length greater than maximum.
                 item length..
55 9 i m         IOdatabase$() item has length greater.
                 than maximum item length..
.
                 m = maximum item length..
                 i = index to item in Data Base..
.
------------------- E X A M P L E --------------------
.
{INPUT:}.
.
DBKEYS Idescription$ = CHR$(25)&&CHR$(1)&&CHR$(5).
.
IODatabase$() = SAMPLE DATA BASE.
.
Iitem$ = "SC02SUPER CATALOGER II".
.
{OUTPUT:}.
.
IODatabase$() = ITEM   VALUE.
                  1    "SC01DSUPER CATALOGER".
                  2    "SC02DSUPER CATALOGER II".
                  3    "SXB1DSUPER EXTENDED BASIC".
                  4    "VT01TVIDEO TITLES I".
                  5    "VT01DVIDEO TITLES I".
                  6    "VT02DVIDEO TITLES II".
                  7    "VT03TVIDEO TITLES III".
                  8    "VT03DVIDEO TITLES III".
                  9    "".
.
Note that SC02D was inserted berfore SXB1D because it
was the first higher key encountered.  The old items
2-8 have now become items 3-9..
.
                        DB.9.0.
..
.                     DBREPL.
.
CALL LINK("DBREPL",IOdatabase$(),Iitem$,Ovector).
.
```

```
This subroutine searches all items in IOdatabase$()
until it finds the first item with a key equal to that
of Iitem$.  Once found, it replaces the old item with
Iitem$.  If an item with an equal key is not found,
IOdatabase$() will not be altered..
.
Ovector values:  1 = Error..
                 2 = Item replaced in IOdatabase$()..
                 3 = Equal item not found in.
                     IOdatabase$()..
.
DBDATA VALUES    DESCRIPTION OF ERROR.
56 1 0 0         DBKEYS improperly/not invoked..
56 3 0 0         Iitem$ length cannot be zero..
56 9 0 m         Iitem$ length greater than maximum.
                 item length..
55 9 i m         IOdatabase$() item has length greater.
                 than maximum item length..
.
                 m = maximum item length..
                 i = index to item in Data Base..
.
------------------- E X A M P L E --------------------
.
{INPUT:}.
.
DBKEYS Idescription$ = CHR$(25)&&CHR$(1)&&CHR$(2).
.
IODatabase$() = SAMPLE DATA BASE.
.
Iitem$ = "SC02SUPER CATALOGER II".
.
{OUTPUT:}.
.
IODatabase$() = ITEM   VALUE.
                  1    "SC02DSUPER CATALOGER II".
                  2    "SXB1DSUPER EXTENDED BASIC".
                  3    "VT01TVIDEO TITLES I".
                  4    "VT01DVIDEO TITLES I".
                  5    "VT02DVIDEO TITLES II".
                  6    "VT03TVIDEO TITLES III".
                  7    "VT03DVIDEO TITLES III".
                  8    "".
.
Note that item 1 has been replaced since both it and
Item$ had a value of "SC" in position 1-2..
.
.
.
.
                        DB.10.0.
..
.                     DBUPDT.
.
CALL LINK("DBUPDT",IOdatabase$(),Iitem$,Ovector).
.
This subroutine searches all items in IOdatabase$()
until it finds an item with either an equal or higher
key than that of Iitem$.  If an equal key is found, it
```

```
replaces the old item with Iitem$.  If a higher key is
found, it inserts Iitem$ into IOdatabase$() moving all
subsequent items down one position.  If all items in
IOdatabase$() have lower key values, Iitem$ will be
placed at the end of IOdatabase$().  This subroutine
assumes that IOdatabase$() has  previously been
sorted..
.
Ovector values:  1 = Error..
                 2 = O.K., Item replaced in.
                     IOdatabase$()..
                 3 = O.K., item inserted in.
                     IOdatabase$()..
.
DBDATA VALUES    DESCRIPTION OF ERROR.
57 1 0 0         DBKEYS improperly/not invoked..
57 3 0 0         Iitem$ length zero..
57 9 0 m         Iitem$ length greater than maximum.
                 item length..
55 9 i m         IOdatabase$() item has length greater.
                 than maximum item length..
.
                 m = maximum item length..
                 i = index to item in Data Base..
.
(examples start on the following page.
..
                      DB.11.0.
..
DBUPDT (Continued).
.
.
---------------- E X A M P L E  # 1 ------------------
.
{INPUT:}.
.
DBKEYS Idescription$ = CHR$(25)&&CHR$(1)&&CHR$(5).
.
IODatabase$() = SAMPLE DATA BASE.
.
Iitem$ = "SC02SUPER CATALOGER (TM)".
.
{OUTPUT:}.
.
IODatabase$() = ITEM   VALUE.
                  1    "SC01DSUPER CATALOGER (TM)".
                  2    "SXB1DSUPER EXTENDED BASIC".
                  3    "VT01TVIDEO TITLES I".
                  4    "VT01DVIDEO TITLES I".
                  5    "VT02DVIDEO TITLES II".
                  6    "VT03TVIDEO TITLES III".
                  7    "VT03DVIDEO TITLES III".
                  8    "".
.
Note that item 1 was replaced..
.
---------------- E X A M P L E # 2 ------------------ .
.
{INPUT:}.
```

```
.
DBKEYS Idescription$ = CHR$(25)&&CHR$(1)&&CHR$(5).
.
IODatabase$() = SAMPLE DATA BASE.
.
Iitem$ = "SC02SUPER CATALOGER II".
.
{OUTPUT:}.
.
IODatabase$() = ITEM   VALUE.
                  1    "SC01DSUPER CATALOGER".
                  2    "SC02DSUPER CATALOGER II".
                  3    "SXB1DSUPER EXTENDED BASIC".
                  4    "VT01TVIDEO TITLES I".
                  5    "VT01DVIDEO TITLES I".
                  6    "VT02DVIDEO TITLES II".
                  7    "VT03TVIDEO TITLES III".
                  8    "VT03DVIDEO TITLES III".
                  9    "".
.
Note that the new item was inserted as item 2 and all
subsequent items were pushed down one position in the
Data Base..
.
                        DB.12.0.
..
.                     DBEQCT.
.
CALL LINK("DBEQCT",Idatabase$(),Iitem$,Ocount,Ovector)
.
This subroutine searches all items in Idatabase$() and
returns a count of all items which have key fields
identical to those of Item$..
.
Ovector values:  1 = Error..
                 2 = Equal item(s) found in.
                     Idatabase$()..
                 3 = No equal items found in.
                     Idatabase$()..
.
DBDATA VALUES    DESCRIPTION OF ERROR.
58 1 0 0         DBKEYS improperly/not invoked..
58 3 0 0         Iitem$ length zero..
58 9 0 m         Iitem$ length greater than maximum.
                 item length..
58 9 i m         IOdatabase$() item has length greater.
                 than maximum item length..
.
                 m = maximum item length..
                 i = index to item in Data Base..
.
------------------- E X A M P L E --------------------
.
{INPUT:}.
.
DBKEYS Idescription$ = CHR$(25)&&CHR$(5)&&CHR$(1).
.
IDatabase$() = SAMPLE DATA BASE.
.
```

```
Iitem$ = "----D".
.
{OUTPUT:}.
.
Ocount = 5 (disk versions exist)..
..
                      DB.13.0.
..
.                     DBFIND.
.
CALL LINK("DBFIND",Idatabase$(),Iitem$,IOpointer,.
     Ovector).
.
This subroutine searches all items in Idatabase$()
starting at IOpointer + 1 and returns IOpointer with
either the next item number in Idatabase$() which has
key fields identical to those of Iitem$ or a value of
zero if no subsequent match was found..
.
Ovector values:  1 = Error..
                 2 = Next equal item found in.
                     Idatabase$()..
                 3 = Equal item not found in.
                     Idatabase$()..
.
DBDATA VALUES    DESCRIPTION OF ERROR.
53 1 0 0         DBKEYS improperly/not invoked..
53 3 0 0         Iitem$ length zero..
53 9 0 m         Iitem$ length greater than maximum.
                 item length..
53 9 i m         Idatabase$() item has length greater.
                 than maximum item length..
.
                 m = maximum item length..
                 i = index to item in Data Base..
.
------------------- E X A M P L E --------------------
.
{INPUT:}.
.
DBKEYS Idescription$ = CHR$(25)&&CHR$(6)&&CHR$(20).
.
IDatabase$() = SAMPLE DATA BASE.
.
Iitem$ = "-----VIDEO TITLES I".
.
IOpointer = 0.
.
{OUTPUT:}.
.
IOpointer = 3 (first occurance in database).
..
                       DB.14.0.
..
DBFIND (Continued).
.
.
---------------- E X A M P L E  # 2 ------------------.
.
```

```
{INPUT:}.
.
DBKEYS Idescription$ = CHR$(25)&&CHR$(6)&&CHR$(20).
.
IDatabase$() = SAMPLE DATA BASE.
.
Iitem$ = "-----VIDEO TITLES I".
.
IOpointer = 3 (same as example #1).
.
{OUTPUT:}.
.
IOpointer = 4 (second occurance in Data base).
..
                      DB.15.0.
..
.                     DBCOMP.
.
CALL LINK("DBCOMP",Iitem1$,Iitem2$,Ovector).
.
This subroutine compares two Data Base formatted items
(i.e., Iitem$1 against Iitem$2) based only on key
fields specified with the DBKEYS subroutine..
.
Ovector values:  1 = Error..
                 2 = Iitem1$ < Iitem2$..
                 3 = Iitem1$ = Iitem2$..
                 4 = Iitem1$ > Iitem2$..
.
DBDATA VALUES    DESCRIPTION OF ERROR.
52 1 0 0         DBKEYS improperly/not invoked..
52 3 1 0         Iitem1$ length zero..
52 3 2 0         Iitem2$ length zero..
52 9 1 m         Iitem1$ length greater than maximum.
                 item length..
53 9 2 m         Iitem2$ length greater than maximum.
                 item length..
.
                 m = maximum item length..
.
---------------- E X A M P L E   # 1 -----------------
.
{INPUT:}.
.
DBKEYS Idescription$ = CHR$(25)&&CHR$(6)&&CHR$(20).
.
Iitem1$ = "VT01TVIDEO TITLES I".
.
IITEM2$ = "VT01DVIDEO TITLES I".
.
{OUTPUT:}.
.
Ovector = 3 (Iitem1$ = Iitem2$).
..
                      DB.16.0.
..
DBCOMP (Continued).
.
.
```

```
---------------- E X A M P L E  # 2 ------------------
.
{INPUT:}.
.
DBKEYS Idescription$ = CHR$(25)&&CHR$(6)&&CHR$(20).
.
Iitem1$ = "SC01DSUPER CATALOGER".
.
IITEM2$ = "VT03VIDEO TITLES III".
.
{OUTPUT:}.
.
Ovector = 2 (Iitem1$ < Iitem2$).
.
.
---------------- E X A M P L E  # 3 ------------------
.
{INPUT:}.
.
DBKEYS Idescription$ = CHR$(25)&&CHR$(6)&&CHR$(20).
.
Iitem1$ = "SXB1DSUPER EXTENDED BASIC".
.
Iitem2$ = "SC01DSUPER CATALOGER".
.
{OUTPUT:}.
.
Ovector = 4 (Iitem1$ > Iitem2$).
..
                      DB.17.0.
..
.                     DBOKEY.
.
CALL LINK("DBOKEY",Idatabase1$(),Odatabase2$(),.
     Ovector).
.
This subroutine transfers all key fields from
Idatabase1$() to Odatabase$2() in such a manner so as
to create one contiguous key field for each output
item.  Although not readily apparent, this subroutine
can also be used to quickly perform the equivalent of
extremely complicated SEG$ statements.  You will have
to experiment with this subroutine to actually get a
good feel for this capability..
.
Ovector values:  1 = Error..
                 2 = O.K..
.
DBDATA VALUES    DESCRIPTION OF ERROR.
59 1 0 0         DBKEYS improperly/not invoked..
.
.
------------------- E X A M P L E --------------------
.
{INPUT:}.
.
DBKEYS Idescription$ = CHR$(25)&&CHR$(1)&&CHR$(5).
.
IDatabase$() = SAMPLE DATA BASE.
```

```
.
{OUTPUT:}.
.
ODatabase2$() = ITEM   VALUE.
                  1    "SC01D".
                  2    "SXB1D".
                  3    "VT01T".
                  4    "VT01D".
                  5    "VT02D".
                  6    "VT03T".
                  7    "VT03D".
                  8    "".
..
                        DB.18.0.
..
......................
```

{S T R I N G A R R A Y}.
{C O N S I D E R A T I O N S}.

SXB String Arrays are considered to include only item one through the first item containing a null string (i.e.,"")in the array. If item one contains a null string, then the array is considered to be "empty" even though other items might contain values other than null strings. If you are using OPTION BASE 1, you cannot access item zero. If you are using OPTION BASE 0 (the default value), you may use item zero for any purpose -- {just remember that it will be ignored by SXB String Array subroutines:}.

```
.
                  {SAMPLE STRING ARRAY}.
.
ITEM    VALUE                   NOTES.
  0     "any value"             Ignored by SXB..
-------------------------------------------------------.
  1     "111 CONSTITUTION"      Only item one through.
  2     "FIRST AND MAPLE"       the first item with a.
  3     "P.O. BOX 415"          null string are.
  4     "2820 S ABINGDON ST"    considered as the.
  5     "APT 43"                String Array by SXB.
  6     "WASHINGTON DRIVE".
  7     "".
-------------------------------------------------------.
  8     "any value"             Ignored by SXB.
  9     "any value".
etc..
.
              STRING ARRAY SUBROUTINES.
.
SAICNT  Count the number of non-null string items in.
        the array..
SACCNT  Count the number of characters used for items.
        in the array..
SAMIN   Determine the length of the shortest non-null.
        string in the array..
SAMAX   Determine the length of the longest non-null.
        string in the array..
SATRNS  Translate specified characters in the array to.
        new ones..
SANCOD  Encode (encrypt) String Array according to.
        password..
SADCOD  Reverse SANCOD process with same password..
SAZAP   Make all non-null strings in the array into.
        null strings..
SACOPY  Copy one String Array to another String Array..
SAVIEW  View String Array on screen with item numbers.
        + lengths..
.
.
.
.
                       SA.2.0.
..
.                      SAICNT.


CALL LINK("SAICNT",Iarray$(),Ocount).
```

        This subroutine counts the number of non-null string items in Iarray$() and returns the value in Ocount..

```
          .
          ---------------- E X A M P L E -------------------.
          .
          {INPUT:}.
          .
          Iarray$() = SAMPLE STRING ARRAY.
          .
          {OUTPUT:}.
          .
          Ocount = 6.
          .
          .
          .
          .
          .
          .                       SACCNT.
          .
          CALL LINK("SACCNT",Iarray$(),Ocount).
          .
```

          This subroutine totals the lenghts of all items in Iarray$() and returns the value in Ocount..

```
          .
          .
          ----------------- E X A M P L E ------------------.
          .
          {INPUT:}.
          .
          Iarray$() = SAMPLE STRING ARRAY.
          .
          {OUTPUT:}.
          .
          Ocount = 82.
          .
          .
          .
                                SA.3.0.
          ..
          .                       SAMIN.
          .
          .
          CALL LINK("SAMIN",Iarray$(),Ocount).
          .
```

          This subroutine checks the length of each non-null string item in Iarray$() to determine the shortest length,which is returned in Olength. If Iarray$() is empty, length will return with a value of zero..

```
          .
          .
          ------------------ E X A M P L E -------------------.
          .
          {INPUT:}.
          .
          Iarray$() = SAMPLE STRING ARRAY.
          .
          {OUTPUT:}.
          .
          Olength = 6.
          .
          .
          .
          .
          .                       SAMAX.
          .
          .
          CALL LINK("SAMAX",Iarray$(),Olength).
          .
          This subroutine checks the length of each non-null string item in Iarray$
() to determine the longest length, which is returned in Olength.  If Iarray$() is
empty, Olength will return with a value of zero..
          .
          .
          ------------------ E X A M P L E -------------------.
          .
          {INPUT:}.
          .
          Iarray$() = SAMPLE STRING ARRAY.
          .
          {OUTPUT:}.
          .
          Olength = 18.
          .
          .

          .
                                 SA.4.0.
          ..
          .                      SATRNS.
          .
          .
          CALL LINK("SATRNS",IOarray$(),Itable$).
          .
          This subroutine translates selected characters in IOarray$() based on
values in Itable$.  The format for Itable$ isas follows:  Each pair of character
in Itable$ represent an item in the table.  The first character of each pair
indicates the character that is to be translated.  The second character indicates
the replacement character. (e.g.If Itable$ were equal to "AQNVJR", each "A" would
be replaced with a "Q", each  "N" with a "V" and each "J" with an "R".).
          .
          {ADDITIONAL MEMORY REQUIREMENTS:}.
          .
          An additional 256 bytes of the 24K portion of memory is
          temporarily required in order to use this subroutine..
          .
          .
          ------------------ E X A M P L E -------------------.
          .
          {INPUT:}.
          .
          IOarray$() = SAMPLE STRING ARRAY.
          .
          Itable$ = " -AaBz".
          .
          {OUTPUT}.
          .
          IOarray$() = ITEM  VALUE.
                         1   "1111-CONSTITUTION".
                         2   "FIRST-&&-MaPLE".
                         3   "P.O.-ZOX-415".
                         4   "2820-S-aZINGDON-ST".
                         5   "aPT-43".
                         6   "WaSHINGTON-DRIVE".
                         7   "".
          .
          Another practical use for this subroutine would be to
          convert normally unused characters to needed binary values which are
required for other SXB subroutines. As an example, suppose you do not need to use
lower case letters "a", "b", "c" but you do need the binary values 1,2 and 3(i.e.,
CHR$(1), CHR$(2) AND CHR$(3)). This can be accomplished with Itable$ equal to
"a"&&CHR$(1)&&"b"&&CHAR $(2)&&"c"&&CHR$(3).  You may consider reserving a special
string array (with a DIM(2)) for converting single strings -- just place the string
in item one and never put any value in item two -- then use SATRNS..
          .
          .
          .
                                SA.5.0.
          ..
          .                     SANCOD.
          .
          CALL LINK("SANCOD",IOarray$(),Ipassword$).
          .
          This subroutine encodes (encrypts) all non-null string items in IOarray$
() so as to be unreadable.  The length of Ipassword$ is directly related to the
complexity of encryption scheme which is used on IOarray$().  Using the same
Ipassword$ with the SADCOD subroutine is required to restore IOarray$() to it's
original condition..
          .
          .
          .
          .
          .                     SADCOD.
          .
          CALL LINK("SADCOD",IOarray$(),Ipassword$).
          .
          This subroutine reverses the encryption process described in the SANCOD
subroutine.  Using the same Ipassword$ on the SANCODed IOarray$() will restore the
string items to their original condition..

                                SA.6.0.
          ..
          .                      SAZAP.
          .
          CALL LINK("SAZAP",IOarray$()).
          .
          This subroutine replaces all non-null string items in
          IOarray$() with null strings..
          .
          .
          ------------------- E X A M P L E --------------------.
          .
          {INPUT:}.
          .
          IOarray$() = SAMPLE STRING ARRAY.
          .
          {OUTPUT:}.
          .
          IOarray$() = ITEM   VALUE.
                         1      "".
                         2      "".
                         3      "".
                         4      "".
                         5      "".
                         6      "".
                         7      "".
          ..
          .                       SACOPY.
          .
          CALL LINK("SACOPY",Iarray1$(),Oarray2$()()).
          .
          This subroutine moves a copy of each item in Iarray1$() to the
corresponding item in Oarray2$()..
          .
          .
          ------------------ E X A M P L E -------------------.
          .
          {INPUT:}.
          .
          Iarray1$() = SAMPLE STRING ARRAY.
          .
          {OUTPUT:}.
          .
          Oarray2$() = ITEM   VALUE.
                         1      "111 CONSTITUTION".
                         2      "FIRST & MAPLE".
                         3      "P.O BOX 415".
                         4      "2820 S ABINGTON ST".
                         5      "APT 43".
                         6      "WASHINGTON DRIVE".
                         7      "".
          .
          .
                                  SA.7.0.
          ..
          .                       SAVIEW.
          .
          CALL LINK("SAVIEW",Iarray$()).
          .
          This subroutine scrolls items from Iarray$() on the screen preceded by a
line of hyphens with two numbers on the right indicating the item number and item
length.  If there is a null string in the first item of Iarray$(), the subroutine
will do nothing. otherwise, it will automatically scroll the first item onto the
bottom of the screen.  As long as the second item of Iarray$() is not a null
string, you can now scroll through Iarray$() by using the Down arrow (FCTN X) to
display items in ascending order or the UP arrow (FCTN E) to display
items in descending order. Press ENTER to exit the subroutine..
          .
          {ADDITIONAL MEMORY REQUIREMENTS:}.
          .
          An aditional 768 bytes of the 24K portion of memory is
          temporarily required in order to use this subroutine..
          .

          ----------------- E X A M P L E ------------------.
          .
          {INPUT:}.
          .
          Iarray$() = SAMPLE STRING ARRAY.
          .
          (momentarily pressing the DOWN arrow key)
          .
          {OUTPUT:}.
          .
          Screen:.
          .
          .
                        ----------------------1--17-.
                        1111 CONSTITUTION.
                        ----------------------2--13-.
                        FIRST & MAPLE.
                        ----------------------3--12-.
                        P.O. BOX 415.
                        ----------------------4--18-.
                        2820 S ABINGDON ST.
                        ----------------------5---6-.
                        APT 43.
                        ----------------------6--16-.
                        WASHINGTON DRIVE.
          .
                                 SA.8.0.
          ..

......................
```

```
                      {S T R I N G}.
              {C O N S I D E R A T I O N S}.
.
SXB strings are the same as regular Extended BASIC
strings..
.
                 {STRING SUBROUTINES}.
.
SMHXBN  Convert readable hexadecimal to binary (two
        hex characters to one binary character ratio).
.
SMBNHX  Convert binary to readable hexadecimal (one
        binary character to two hex characters ratio).
.
SMFIX   Fix the length of a string..
.
SMSNIP  Delete specified number of characters from a
        string..
.
SMTRIM  Delete specified trailing characters from a
        string..
.
SMSWAP  Swap the values of two strings..
.
SMRDUC  Reduces length of string by changing each.
        group of 1-16 continuous blanks/zeros to 1.
        byte..
.
SMXPND  Restores string created by SMRDUC to it's
        original condition..
..
                    SM.2.0.
..
.                    SMHXBN.
.
.
CALL LINK("SMHXBN",Ihexadecimal$,Obinary$).
.
This subroutine converts a readable hexadecimal string
(Ihexadecimal$)  to  a   binary   string   (Obinary$)..
..
CAUTION: {Only hexadecimal characters (0, 1, 2, 3, 4,}
{5, 6, 7, 8, 9, A, B, C, D, E and F) are valid input}
..
for Ihexadecimal$.  Input of invalid characters will}
produce unpredictable results.}.
.
{ADDITIONAL MEMORY REQUIREMENTS:}.
.
An addditional 256 bytes of the 24K portion of memory
is temporarily required in order to use this
subroutine..
.
.
------------------- E X A M P L E --------------------.
.
{INPUT:}.
.
```

```
Ihexadecimal$ = "4142434445464748"   (length 16).
.
{OUTPUT:}.
.
Obinary$ ="ABCDEFGH"   (length 8).
.
.
.
.                       SMBNHX.
.
.
CALL LINK("SMBNHX",Ibinary$,Ohexadecimal$).
.
This subroutine converts a binary string (Ibinary$) to
a readable hexadecimal string (Ohexadecimal$)..
..
CAUTION: {The output string will be twice the length}
{of the input string.  If the input string is greater}
{than 127 characters in length, it will be shortened
by} {128 characters!}.
..
.
------------------- E X A M P L E --------------------.
.
{INPUT:}.
.
Ibinary$ = "ABCDEFGH"   (length 8).
.
{OUTPUT:}.
.
Ohexadecimal$ = "4142434445464748"   length 16).
..
                      SM.3.0.
..
.                       SMFIX.
.
.
CALL LINK("SMFIX",IOstring$,Ilength).
.
This subroutine fixes the lingth of IOstring$.  If it
is longer on input, it will be truncated on output.  If
it is shorter on input, it will be expanded to the
specified length with trailing blanks..
.
.
----------------- E X A M P L E  # 1 ----------------.
.
{INPUT:}.
.
IOstring$ = "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
.
Ilength = 15
.
{OUTPUT:}.
.
IOstring$ = "ABCDEFGHIJKLMNO".
.
.
----------------- E X A M P L E  #2 ----------------.
```

```
.
.
{INPUT:}.
.
IOstring$ = "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
.
Ilength = 27.
.
{OUTPUT}.
.
IOstring$ = "ABCDEFGHIJKLMNOPQRSTUVWXYZ ".
.
Note the extra blank following the z..
..
                      SM.4.0.
..
.                      SMSNIP.
.
CALL LINK("SMSNIP",IOstring$,Ilength).
.
This subroutine "snips" off Ilength characters from the
end of IOstring$.  If IOstring$ is less than Ilength
characters in length, IOstring$ will be returned as a
null string..
.
.
---------------- E X A M P L E  # 1 ---------------.
.
{INPUT:}.
.
IOstring$ = "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
.
Ilength = 5.
.
{OUTPUT:}.
IOSTRING$ = "ABCDEFGHIJKLMNOPQRSTU".
.
.
---------------- E X A M P L E  # 2 ---------------.
.
{INPUT:}.
.
IOstring$ = "ABC".
.
ILENGTH = 5.
.
{OUTPUT:}.
.
IOstring$ = "".
..
                      SM.5.0.
..
.                      SMTRIM.
.
.
CALL LINK("SMTRIM",IOstring$,Itrimlist$).
.
This subroutine deletes all trailing characters from
IOstring$ that match any of the characters in
```

```
Itrimlist$.  Once it finds any character that does not
have a match in Itrimlist$, the truncation process
stops..
.
------------------ E X A M P L E -------------------.
.
{INPUT:}.
.
IOstring$ = "ABCXYZABCABCABCXYZZZZYXZYYXZ".
.
Itrimlist$ = "XYZ".
.
{OUTPUT:}.
.
IOstring$ = "ABCXYZABCABCABC".
.
Note that the letters "XYZ" are still left in the first
part of the string..
.SP 31.
                      SM.6.0.
..
.                      SMSWAP.
.
.
CALL LINK("SMSWAP",IOstring1$,IOstring2$).
.
This routine swaps the values in IOstring1$ and
IOstring2$..
.
.
------------------ E X A M P L E -------------------.
.
{INPUT:}.
.
IOstring1$ = "ABCDEFGHIJKLMN".
.
IOstring2$ = "OPQRSTUVWXYZ".
.
{OUTPUT:}.
.
IOstring1$ = "OPQRSTUVWXYZ".
.
IOstring2$ = "ABCDEFGHIJKLMN".
..
                      SM.7.0
..
.                      SMRDUC.
.
.
CALL LINK("SMRDUC",Iregular$,Oreduced$).
.
This routine reduces the length of Iregular$ by
changing each group of 1-16 continuous blanks or zeros
..
to 1 byte.  CAUTION: {Since the binary values 0-31 are}
{used to designate the compacted data, Iregular$ must}
{never contain these characters (i.e., CHRS(0-31)) or}
{the string will not be able to be reconstructed by
the} {SMXPND routine.
```

```
.
.
.                       SMXPND.
.
.
CALL LINK("SMXPND",Ireduced$,Oregular$).
.
This subroutine reverses the process of the SMRDUC
subroutine..
..
                    SM.8.0.
..

......................
```

```
                     {I N T E G E R}.
              {C O N S I D E R A T I O N S}.
.
SXB allows for the storage of four times as many
integers (in the range of -32,768 to +32,767) in a
numeric array than in Extended BASIC (i.e., a numeric
array such as DIM INTEGER(250) can store 1,000
integers).  Each item in a numeric array would
normally hold one Radix-100 notated number in Extended
BASIC.  The SXB Integer subroutines allow up to four
integers to occupy the same amount of storage.
Referencing items in the SXB Integer Array (henceforth
referred to as Integer Array) will be acomplished in
the following manner:  IOarray(),Ipointer.  This will
replace the  format you are  normally used to  working
..
with (i.e., IOarray(Ipointer)).  CAUTION:
{Inadvertent} {input of numbers outside the range of
-32,768 to} {+32,767 will produce unpredictable
results!}.
..
.
{NUMERIC ARRAY}              {SXB INTEGER ARRAY}.
{                      }     {                      }.
{|         1          |}     {|  1 |  2 |  3 |  4 |}.
{|         2          |}     {|  5 |  6 |  7 |  8 |}.
{|         3          |}     {|  9 | 10 | 11 | 12 |}.
{|         4          |}     {| 13 | 14 | 15 | 16 |}.
.
                  {INTEGER SUBROUTINES}.
.
INPAK4   Pack four integers into one numeric
variable..
INUPK4   Unpack four integers from one numeric.
         variable..
INPAK    Pack one integer into an Integer Array cell..
INUPK    Unpack one integer from an Integer Array.
         Cell..
INZERO   Zero an Integer Array cell..
ININC    Add 1 to the integer in an Integer Array.
         cell..
INDEC    Subtract 1 from the integer in an Integer.
         Array cell..
INPOS    Make the integer in an Integer Array cell.
         positive..
INNEG    Make the integer in an Integer Array cell.
         negative..
INCHNG   Change the sign of the integer in an Integer.
         Array cell..
INGETS   Move (portion of) Integer Array to a string.
         variable..
INPUTS   Move string variable to (a portion of).
         Integer Array..
INvADD   Add integers..
INvSUB   Subtract integers..
INvMPY   Multiply integers..
INvDIV   Divide integers..
.
.
```

```
                             IN.2.0.
..
.                      INPAK4.
.
CALL LINK("INPAK4",Onumericvar,Iinteger1,Iinteger2,.
     Iinteger3,Iinteger4).
.
This subroutine packs Iinteger1, Iinteger2, Iinteger3,
and Iinteger4 into Onumericvar.  Onumericavar may be
part of an Integer Array. If it is, Array(2) actually
references Integer Array, 5-8, etc.  If Iinteger1,
Iinteger2, Iinteger3, or Iinteger4 is not a whole
number, it will be truncated to one..
.
.
.                      INUPK4.
.
CALL LINK("INUPK4",Inumericvar,Ointeger1,Ointeger2,.
     Ointeger3,Ointeger4).
.
This subroutine reverses the process of the INPAK4
subroutine..
.
.
.                       INPAK.
.
CALL LINK("INPAK",Oarray(),Ipointer,Iinteger).
.
This subroutine packs Iinteger into the specified cell
of Oarray(),Ipointer..
.
.
.                       INUPK.
.
CALL LINK("INUPK",Iarray(),Ipointer,Ointeger).
.
This subroutine reverses the process of the INPAK
subroutine..
.
.
                             IN.3.0.
..
.                      INZERO.
.
CALL LINK("INZERO",IOarray(),Ipointer).
.
This subroutine zeroes the integer in the specified
cell of IOarray(),Ipointer..
.
.
.                       ININC.
.
CALL LINK("ININC",IOarray(),Ipointer).
.
This subroutine adds 1 to the integer in the specified
cell of IOarray(),Ipointer..
.
.
.                       INDEC.
```

.
CALL LINK("INDEC",IOarray(),Ipointer).
.
This subroutine subtracts 1 from the integer in the
specified cell of IOarray(),Ipointer..
.
.
.   INPOS.
.
CALL LINK("INPOS",IOarray(),Ipointer).
.
This subroutine makes the integer in the specified
cell of IOarray(),Ipointer positive..
.
.
.   INNEG.
.
CALL LINK("INNEG",IOarray(),Ipointer).
.
This subroutine makes the integer in the specified
cell of IOarray(),Ipointer negative..
.
.
.   INCHNG.
.
CALL LINK("INCHNG",IOarray(),Ipointer).
.
This subroutine changes the sign of the integer in the
specified cell of IOarray(),Ipointer..
.
.
                           IN.4.0.
..
.   INGETS.
.
CALL LINK("INGETS",Iarray(),Ipointer,Icount,Ostring$).
.
This subroutine retrieves compacted Integer Array cell
values, starting at Iarray(),Ipointer and ending at
Iarray(),Ipointer+Icount-1, into Ostring$. This
produces an easy method of saving an Integer Array in
an output file for later reuse. The maximum Icount is
127 since this would cause a maximum even length
string (i.e., 254 characters) as each cell takes up
.RM +2.
two characters. CAUTION: {When retrieving strings of}
{this type you must use the LINPUT statement.
Failure} {to follow this advice will possibly result
in the loss} {of data when it is retrieved.}.
..
.
.
.
.   INPUTS.
.
CALL LINK("INPUTS",IOarray(),Ipointer,Istring$).
.
This subroutine reloads the Integer Array cells from
Istring$ starting at IOarray(),Ipointer and continuing

```
based on the length of Istring$.  Every two characters
fills one Integer Array cell..
.
.
.
.
                           IN.5.0.
..
.                      INvADD.
.
CALL LINK("IN0ADD",0,Iaddend1,Iaddend2,Osum).
CALL LINK("IN0ADD",0,Iaddend1,Iaddend2&&Osum).
.
CALL LINK("IN1ADD",IOintegerarray(),Iaddend1,Iaddend2,.
     Isumpointer).
CALL LINK("IN1ADD",0,Iaddend1,Iaddend2&&Osum).
.
CALL LINK("IN2ADD",Iintegerarray(),Iaddend1,.
     Iaddend2pointer,Osum).
CALL LINK("IN2ADD",IOintegerarray(),Iaddend1,.
     Iaddend2pointer&&Isumpointer).
.
CALL LINK("IN3ADD",IOintegerarray(),Iaddend1,.
     Iaddend2pointer,Isumpointer).
CALL LINK("IN3ADD",IOintegerarray(),Iaddend1,.
     Iaddend2pointer&&Isumpointer).
.
CALL LINK("IN4ADD",Iintegerarray(),Iaddend1pointer,.
     Iaddend2,Osum).
CALL LINK("IN4ADD",Iintegerarray(),Iaddend1pointer,.
     Iaddend2&&Osum).
.
CALL LINK("IN5ADD",IOintegerarray(),Iaddend1pointer,.
     Iaddend2,Isumpointer).
CALL LINK("IN5ADD",Iintegerarray(),Iaddend1pointer,.
     Iaddend2&&Osum).
.
CALL LINK("IN6ADD",Iintegerarray(),Iaddend1pointer,.
     Iaddend2pointer,Osum).
CALL LINK("IN6ADD",IOintegerarray(),Iaddend1pointer,.
     Iaddend2pointer&&Isumpointer).
.
CALL LINK("IN7ADD",IOintegerarray(),Iaddend1pointer,.
     Iaddend2pointer,Isumpointer).
CALL LINK("IN7ADD",IOintegerarray(),Iaddend1pointer,.
     Iaddend2pointer&&Isumpointer).
.
.
These subroutines provide the capability to either add
two integers together to produce a sum or add one
integer to another.  All possible combinations of
interaction between numeric variables and Integer
Array cell variables have been provided for your
programming convenience..
.
.
           400          addend2.
          {+ 50}        addend1.
           450          sum.
```

```
.
.
See table on page IN.10.0 for further clarification..
.
                          IN.6.0.
..
.                    INvSUB.
.
CALL LINK("IN0SUB",0,Isubtrahend,Iminuend,Odifference).
CALL LINK("IN0SUB",0,Isubtrahend,Imenuend&&Odifference) .
.
CALL LINK("IN1SUB",IOintegerarray(),Isubtrahend,.
     Iminuend,Idifferencepointer).
CALL LINK("IN1SUB",0,Isubtrahend,Iminuend&&Odifference) .
.
CALL LINK("IN2SUB",Iintegerarray(),Isubtrahend,.
     Iminuendpointer,Odifference).
CALL LINK("IN2SUB",IOintegerarray(),Isubtrahend,.
     Iminuendpointer&&Idifferencepointer).
.
CALL LINK("IN3SUB",IOintegerarray(),Isubtrahend,.
     Iminuendpointer,Idifferencepointer).
CALL LINK("IN3SUB",IOintegerarray(),Isubtrahend,.
     Iminuendpointer&&Idifferencepointer).
.
CALL LINK("IN4SUB",Iintegerarray(),Isubtrahend pointer,.
     Iminuend,Odifference).
CALL LINK("IN4SUB",Iintegerarray(),Isubtrahendpointer,.
     Iminuend&&Odifference).
.
CALL LINK("IN5SUB",IOintegerarray(),.
     Isubtrahendpointer,Iminuend,Idifferencepointer).
CALL LINK("IN5SUB",Iintegerarray(),Isubtrahendpointer,.
     Iminuend&&Odifference).
.
CALL LINK("IN6SUB",Iintegerarray(),Isubtrahendpointer,.
     Iminuendpointer,Odifference).
CALL LINK("IN6SUB",IOintegerarray(),.
     Isubtrahendpointer,.
     Iminuendpointer&&Idifferencepointer).
.
CALL LINK("IN7SUB",IOintegerarray(),.
     Isubtrahendpointer,Iminuendpointer,.
     Idifferencepointer).
CALL LINK("IN7SUB",IOintegerarray(),.
     Isubtrahendpointer,.
     Iminuendpointer&&Idifferencepointer).
.
These subroutines provide the capability to either
subtract one integer from another to produce a
difference or subtract one integer from another.  All
possible combinations of interaction between numeric
variables and Integer Array cell variables have been
provided for your programming convenience..
.
           400             minuend.
          {- 50}          subtrahend.
           350            difference.
.
```

See table on page IN.10.0 for further clarification..
.

..
. INvMPY.
.

```
CALL LINK("IN0MPY",0,Imultiplier,Imultiplicand,.
     Oproduct).
CALL LINK("IN0MPY",0,Imultiplier,.
     Imultiplicand&&Oproduct).
.
CALL LINK("IN1MPY",IOintegerarray(),Imultiplier,.
     Imultiplicand,Iproductpointer).
CALL LINK("IN1MPY",0,Imultiplier,.
      Imultiplicand&&Oproduct) .
.
CALL LINK("IN2MPY",Iintegerarray(),Imultiplier,.
     Imultiplicandpointer,Oproduct).
CALL LINK("IN2MPY",IOintegerarray(),Imultiplier,.
     Imultiplicandpointer&&Iproductpointer).
.
CALL LINK("IN3MPY",IOintegerarray(),Imultiplier,.
     Imultiplicandpointer,Iproductpointer).
CALL LINK("IN3MPY",IOintegerarray(),Imultiplier,.
     Imultiplicandpointer&&Iproductpointer).
.
CALL LINK("IN4MPY",Iintegerarray(),Imultiplierpointer,.
     Imultiplicand,Oproduct).
CALL LINK("IN4MPY",Iintegerarray(),Imultiplierpointer,.
     Imultiplicand&&Oproduct).
.
CALL LINK("IN5MPY",IOintegerarray(),.
     Imultiplierpointer,Imultiplicand ,Iproductpointer).
CALL LINK("IN5MPY",Iintegerarray(),Imultiplierpointer,.
     Imultiplicand&&Oproduct).
.
CALL LINK("IN6MPY",Iintegerarray(),Imultiplierpointer,.
     Imultiplicandpointer,Oproduct).
CALL LINK("IN6MPY",IOintegerarray(),.
     Imultiplierpointer,.
     Imultiplicandpointer&&Iproductpointer).
.
CALL LINK("IN7MPY",IOintegerarray(),.
     Imultiplierpointer,Imultiplicandpointer,.
     Iproductpointer).
CALL LINK("IN7MPY",IOintegerarray(),.
     Imultiplierpointer,.
     Imultiplicandpointer&&Iproductpointer).
```

These subroutines provide the capability to either multiply two integers together to produce a product or multiply one integer by another. All possible combinations of interaction between numeric variables and Integer Array cell variables have been provided for your programming convenience..

```
           400            multiplicand.
          {x 50}          multiplier.
        20,000            product.
```

See table on page IN.10.0 for further clarification..
.

```
..
.                      INvDIV.
.
CALL LINK("IN0DIV",0,Idivisor,Idividend, Oquotient).
CALL LINK("IN0DIV",0,Idivisor,Idividend&&Oquotient).
.
CALL LINK("IN1DIV",IOintegerarray(),Idivisor,.
     Idividend,Iquotientpointer).
CALL LINK("IN1DIV",0,Idivisor,.
      Idividend&&Oquotient).
.
CALL LINK("IN2DIV",Iintegerarray(),Idivisor,.
     Idividendpointer,Oquotient).
CALL LINK("IN2DIV",IOintegerarray(),Idivisor,.
     Idividendpointer&&Iquotientpointer).
.
CALL LINK("IN3DIV",IOintegerarray(),Idivisor,.
     Idividendpointer,Iquotientpointer).
CALL LINK("IN3DIV",IOintegerarray(),Idivisor,.
     Idividendpointer&&Iquotientpointer).
.
CALL LINK("IN4DIV",Iintegerarray(),Idivisorpointer,.
     Idividend,Oquotient).
CALL LINK("IN4DIV",Iintegerarray(),Idivisorpointer,.
     Idividend&&Oquotient).
.
CALL LINK("IN5DIV",IOintegerarray(),Idivisorpointer,.
     Idividend,Iquotientpointer).
CALL LINK("IN5DIV",Iintegerarray(),Idivisorpointer,.
     Idividend&&Oquotient).
.
CALL LINK("IN6DIV",Iintegerarray(),Idivisorpointer,.
     Idividendpointer,Oquotient).
CALL LINK("IN6DIV",IOintegerarray(),Idivisorpointer,.
     Idividendpointer&&Iquotientpointer).
.
CALL LINK("IN7DIV",IOintegerarray(), Idivisorpointer,.
     Idividendpointer,Iquotientpointer).
CALL LINK("IN7DIV",IOintegerarray(),Idivisorpointer,.
     Idividendpointer&&Iquotientpointer).
.
```

These subroutines provide the capability to either divide one integer into another to produce a quotient or to divide one integer into another. All possible combinations of interaction between numeric variables and Integer Array cell variables have been provided for your programming convenience..

.
.

```
           400          dividend.
          {/ 50}          divisor.
             8          quotient.
```

.
See table on page IN.10.0 for further clarification..
.
.

..
Note concerning INvADD, INvSUB, INvMPY, and INvDIV:.
.
.
There is a definite scheme to the assignment of the
value "v" to each of these subroutine names. It is
best explained with the following table:.
.
.
.

{v	addend1 or subtrahend or multiplier or divisor	addend2 or minuend or multiplicand or dividend	sum or. difference or. product or. quotient }.
0	numeric variable	numeric variable	numeric. variable.
1	numeric variable	numeric variable	Integer. Array. pointer.
2	numeric variable	Integer. Array pointer	numeric. variable.
3	numeric variable	Integer Array pointer	Integer. Array. pointer.
4	Integer. Array pointer	numeric varaible	numeric. variable.
5	Integer Array pointer	numeric variable	Integer. Array. pointer.
6	Integer Array pointer	Integer. Array pointer	numeric. variable.
7	Integer Array pointer	Integer Array pointer	Integer. Array. pointer.

.
.
The last parameter is always optional in which case
the preceeding parameter doubles as itself and the
"result" field..
.
.
.

6. VDP subroutines

6.1. Video Display Processor considerations

SXB Video Display Processor subroutines deal with most of the aspects of the computer that affect what you see on the screen. The screen (in Extended Basic) is divided into 24 rows, each containing 32 screen characters (8 Ã— 8 pixels). SXB allows you to define a "window" in that 24 Ã— 32 screen area. All SXB video subroutines will affect only the current window portion of the screen. By default SXB begins with the window set to ignore the leftmost two columns and rightmost two columns of the screen. Some SXB video subroutines indicate an optional last parameter of zero. Using this parameter will cause the window to
be blanked before the subroutine does anything else.
CAUTION: If you are not getting the expected results. first check to insure that you properly defined the window.

6.2. Video Display Processor subroutines

VMWNDW Define window for activity area on the screen.
VMBLNK Move spaces to the window.
VMTYPE Allows key entry on data directly to the screen with editing and cursor movement.
VMREAD Read data from the window and place in string variable.
VMLIST List information in the window based on string with delimiters between data.
VMDRAW Draw string data on the screen based on data delimited by row and column addresses or direction indicator.
VMPCdd Put string on screen in any one of eight different directions.
VMGCdd Get string from screen in any one of eight different directions.
VMGDEF Get pattern definition of up to 31 characters.
VMPDEF Put (define) up to 31 character patterns.
VMLCL Redefine lower case letters in a typewriter style.
VMFORM Redefine specified characters as lines for creating forms on the screen.
VMCOLR Change character set foreground/background colors.
VMSCOL Change sprite colors.

6.3. VMWNDW

CALL LINK("VMWNDW", Ifirstrow , Ifirstcolumn , Ilastrow , Ilastcolumn)
This subroutine defines the current window for screen activity with SXB Video subroutines.
CAUTION: If an incorrect specification in given, the subroutine will change the window to reflect the startup default values:
Ifirstrow = 1
Ifirstcolumn = 3
Ilastrow = 24
Ilastcolumn = 30
For the purpose of illustration, all examples displaying screen output will have the window set as follows:
Ifirstrow = 3
Ifirstcolumn = 3
Ilastrow = 22
Ilastcolumn = 30
This will produce a frame around the window two bytes in width. Additionally, all of these same examples will have had the cursor character (CHR$(30)) moved to all position prior to calling the SXB subroutine and each subroutine will have utilized the last parameter feature to blank out the window.

6.4. VMBLNK

CALL LINK ("VMBLNK")
This subroutine causes the current window to be blanked out on the screen.

6.5. VMTYPE

CALL LINK("VMTYPE"[, Ivalidlist$ [,0]])
This subroutine allows free-form typing within the current window. Optionally, if Ivalidlist$ is present or a null string, the subroutine will only accept characters

specified in the list. Adding a second parameter will cause the window to be cleared before typing begins. The following keys are used for cursor movement:

FCTN	Action
S	Left
D	Right
E	Up
B	Down
1 (DELETE)	No movement (deletes 1 char)
2 (INSERT)	No movement (insert mode)
3 (ERASE)	Home + clears window
4 (CLEAR)	Left Margin
5 (BEGIN)	Center
6 (PROC'D)	Right Margin
7 (AID)	Left on next line
8 (REDO)	Home
9 (BACK)	Home
= (QUIT)	Center

Pressing ENTER will cause the subroutine to exit.

6.6. VMREAD

CALL LINK("VMREAD", Ostring$)

This subroutine retrieves all characters (up to a maximum of 255) from the current window into Ostring$.

To retrieve more than 255 characters, redefine the window and perform VMREAD multiple corresponding times.

6.7. VMLIST

CALL LINK("VMLIST", Idata$ [,0])

This subroutine allows data to be listed onto the screen starting at Home (upper left corner) of the current window. Idata$ contains the data to be listed on each line separated by a delimiter character which is specified in the first character of Idata$.

6.7.1. Example

INPUT:

Idata$ = " NOW IS THE TIME FOR SXB"

6.8. VMDRAW

CALL LINK("VMDRAW", Idata$ [,0])

This subroutine allows quick placement of data on the screen in a contiguous and/or non-contiguous fashion. The default beginning position is Home within the window with a direction of RIGHT. The data to be placed on the screen is interspersed with instructions on direction change or jump to a new row and column l ocation. The format for these instructions is as follows:

ControlCharacter	Instruction
CHR$(0)	Change direction to UP
CHR$(1)	Change direction to UP and RIGHT
CHR$(2)	Change direction to RIGHT
CHR$(3)	Change direction to DOWN and RIGHT
CHR$(4)	Change direction to DOWN
CHR$(5)	Change direction to DOWN and LEFT
CHR$(6)	Change direction to LEFT
CHR$(7)	Change direction to UP and LEFT
CHR$(8)&CHR$(row)&CHR$(column)	Jump to a new row and a new column

6.8.1. Example

INPUT:

```
Idata$ =  CHR$(8)&CHR$(10)&CHR$(7)&"ABC"
&CHR$(3)&"DEF"
&CHR$(4)&"GHI"
&CHR$(5)&"JKL"
&CHR$(6)&"MNO"
&CHR$(7)&"PQR"
&CHR$(0)&"STU"
```

&CHR$(1)&"VWX"

6.9. VMPCdd

```
CALL LINK("VMPCU", Irow , Icolumn , Istring$ [,0])
CALL LINK("VMPCUR", Irow , Icolumn , Istring$ [,0))
CALL LINK("VMPCR", Irow , Icolumn , Istring$ [,0])
CALL LINK("VMPCDR", Irow , Icolumn , Istring$ [,0))
CALL LINK("VMPCD", Irow , Icolumn , Istring$ [,0))
CALL LINK("VMPCDL", Irow , Icolumn , Istring$ [,0])
CALL LINK("VMPCL", Irow , Icolumn , Istring$ [,0])
CALL LINK("VMPCUL", Irow , Icolumn , Istring$ [,0))
```

These eight subroutines allow for putting Istring$ on the screen in eight different directions. Irow and Icolumn may be either positive or negative numbers. Depending on where they begin and the direction that they are "putting" the characters from Istring$, only those which correspond to coordinates within the current window are actually moved to the screen. All others are ignored.
The scheme to the assignment of the value "dd" to each of 'these subroutine names is as follows:

dd	Direction
U	up
UR	up and right
R	right
DR	down and right
D	down
DL	down and left
L	left
UL	up and left

6.9.1. Examples

```
INPUT:
Irow = 12
Icolumn = 16
Istring$ = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

6.10. VMGCdd

```
CALL LINK("VMGCU", Irow , Icolumn , Ilength , Ostring$ [,0))
CALL LINK("VMGCUR", Irow , Icolumn , Ilength , Ostring$ [,0))
CALL LINK("VMGCR", Irow , Icolumn , Ilength , Ostring$ [,0])
CALL LINK("VMGCDR", Irow , Icolumn , Ilength , Ostring$ [,0])
CALL LINK("VMGCD", Irow , Icolumn , Ilength , Ostring$ [,0))
CALL LINK("VMGCDL", Irow , Icolumn , Ilength , Ostring$ [,0])
CALL LINK("VMGCL", Irow , Icolumn , Ilength , Ostring$ [,0])
CALL LINK("VMGCUL", Irow , Icolumn , Ilength , Ostring$ [,0])
```

These eight subroutines allow for getting Ostring$ from the screen in eight different directions. Irow and Icolumn may be either positive or negative numbers. Depending on where they begin and the direction that they are "getting" the characters from the screen, only those which correspond to coordinates within the current window are actually moved to Ostring$. Any locations which are outside the window will be returned to Ostring$ as null characters (i.e., CHR$(0)).
The scheme to the assignment of the value "dd' to each of these subroutine names is as follows:

dd	Direction
U	up
UR	up and right
R	right
DR	down and right
D	down
DL	down and left
L	left
UL	up and left

6.10.1. Example

INPUT:

Irow = 12
Icolumn = 16
Ilength = 5
Screen = Same condition as with corresponding examples from VMPCdd.
OUTPUT:
Ostring$ = "ABCDE"

6.11. VMGDEF

CALL LINK("VMGDEF", Iasciivalue , Icount , Obinary$)

This subroutine allows for the simultaneous retrieval of up to 31 character pattern definitions. Iasciivalue is the first character pattern definition retrieved. Each character pattern definition will occupy eight characters in Obinary$. This provides an easy means of storing character pattern definitions while the characters are temporarily being redefined for other uses.

CAUTION: There is no error checking to determine if you are retrieving definitions within the normal 14 characters sets allowed in TI Extended Basic.

6.12. VMPDEF

CALL LINK("VMPDEF", Iasciivalue , Ibinary$)

This subroutine allows for the redefinition of up to 31 character patterns simultaneously starting at Iasciivalue and continuing until Ibinary$ is exhausted.

CAUTION: There is no error checking to determine if you are redefining characters outside the normal range. Failure to keep within the normal 14 character sets of TI Extended Basic may cause unpredictable results.

6.13. VMLCL

CALL LINK("VMLCL")

This subroutine redefines all the lower case alphabetic characters so that they will appear (when used on the screen) similar to lower case typewriter letters. You may want to save the original lower case character definitions with the VMGDEF subroutine if you intend to restore them later in the same program.

6.14. VMFORM

CALL LINK("VMFORM", Iasciivalue)

This subroutine will redefine up to 11 characters beginning at Iasciivalue and continue until either all 11 characters have been redefined or the ASCII character 143 has been redefined. The characters that are redefined can then be used to design forms on the screen. The order of character redefinition is as follows:

Character	New definition
Iasciivalue	Upper left corner
Iasciivalue+ 1	Horizontal line
Iasciivalue+ 2	Upper right corner
Iasciivalue+ 3	Vertical line
Iasciivalue+ 4	Lower right corner
Iasciivalue+ 5	Lower left corner
Iasciivalue+ 6	Left "T' intersection
Iasciivalue+ 7	Right "T" intersection
Iasciivalue+ 8	Top "T" intersection
Iasciivalue+ 9	Bottom "T" intersection
Iasciivalue+ 10	Vertical line intersecting horizontal line.

6.15. VMCOLR

CALL LINK("VMCOLR", Itable$)

This subroutine changes character set foreground/ background color combinations based on the information contained in Itable$. The changes always start with character set zero (which is used for the cursor and edge characters) and continue for an equal number of character sets as there are items in Itable$. The format for Itable$ is as follows: Each pair of characters in Itable$ (e.g., CHR$(foreground color)&CHR$(background color)) represent an item in the table. A value of zero will cause the associated color to remain unchanged.

CAUTIONS: An incomplete trailing Itable$ item will be ignored by the subroutine. This subroutine does not check to see if you have exceeded past character set 14 with Itable$. Doing so may cause unpredictable results.

Color codes

Value	Color
1	Transparent
2	Black
3	Medium Green
4	Light Green
5	Dark Blue
6	Light Blue
7	Dark Red
8	Cyan
9	Medium Red
10	Light Red
11	Dark Yellow
12	Light Yellow
13	Dark Green
14	Magenta
15	Gray
16	White

6.15.1. Example 1

INPUT:

```
Itable$ = CHR$(0)&CHR$(0)&RPT$(CHR$(16)&CHR$(5),14)
```

OUTPUT:

Character sets 1-14 will be changed to having a foreground color of White and a background color of Dark Blue. Character set zero remains unchanged.

6.15.2. Example 2

INPUT:

```
Itable$ = RPT$(CHR$(9)&CHR$(0),15)
```

OUTPUT:

The foreground color for all character sets is changed to Medium Red without changing any of the background colors.

6.15.3. Example 3

INPUT:

```
Itable$ = RPT$(CHR$(0),10)&RPT$(CHR$(11)&CHR$(3),3)
```

OUTPUT:

For character sets 5, 6 and 7, the foreground color is changed to Dark Yellow and the background color is changed to Medium Green.

6.16. VMSCOL

```
CALL LINK("VMSCOL", Itable$ )
```

This subroutine changes sprite colors based on the information in Itable$. The changes always start with sprite # 1 and continue for an equal number of sprites as there are items in Itable$. The format for Itable$ is as follows: Each character in Itable$ (e.g., CHR$(color)) represents an item in the table. A value of zero will cause the associated sprite color to remain unchanged.

Color codes

Value	Color
1	Transparent
2	Black
3	Medium Green
4	Light Green
5	Dark Blue
6	Light Blue
7	Dark Red
8	Cyan
9	Medium Red
10	Light Red
11	Dark Yellow
12	Light Yellow
13	Dark Green
14	Magenta
15	Gray

16 White

6.16.1. Example

INPUT:

```
Itable$ = RPT$(CHR$(0),5)&RPT$(CHR$(11),8)
```

OUTPUT:

Sprites 1-5 remain unchanged while the colors for sprites 6-13 (the next eight) will have their color changed to dark yellow.

.   BANNER.
.
CALL LINK("BANNER",Oarray$(),Ititle$).
.
This subroutine allows for the retrieval of character pattern definitions in such a way so as to provide each bit in an "on" status to be represented in the strings of Oarray$() as asterisks (i.e., "*")and those in an "off" status as blanks. Oarray$() will be returned with items 1-8 representing the eight rows of pixels which would normally be used to display the characters of Ititle$. Ititle$ may be a maximum of 31 characters long. .
..
.
CAUTION: {Any Ititle$ length greater than 31 characters} {will be reduced to 31 characters.}.
.2.
.
.
.
------------------- E X A M P L E
.
{INPUT:}.
.
Ititle$ = "SXB".
.
{OUTPUT:}.
.

```
Oarray$():  ITEM    VALUE.
              1     "                        ".
              2     "  ***    *   *   ****   ".
              3     " *   *   *   *    *  *  ".
              4     " *        * *     *  *  ".
              5     "  ***      *      ***   ".
              6     "     *    * *     *  *  ".
              7     " *   *   *   *    *  *  ".
              8     "  ***    *   *   ****   ".
```

.SP 18.
                       MISC.2.0.
..
.   KEY1
.
.
.
CALL LINK("KEY1",Icharacterlist$,Opointer).
.
.
This subroutine will go into a wait loop untill one of the characters in Icharacterlist$ is depressed at which time the subroutine will return with Opointer indicating the character. If Icharacterlist$ is given as a null string (i.e., ""), the value "YNyn" will be automatically substituted..
.
------------------- E X A M P L E --------------------.
.
.
{INPUT:}.

```
.
Icharacterlist$ = "IDRU".
.
The key "R" is depressed..
.
{OUTPUT:}.
.
OPOINTER = 3.
..
                      MISC.3.0.
..
.                    PARMSV.
.
.
CALL LINK("PARMSV",Iparameter$).
.
This subroutine temporarily stores Iparameter$ so that
it can be retrieved in the next SXB program.  The
parameter must be retrieved immediately or it will be
inadvertently destroyed..
.
.
.
.
.
.                    PARMGT.
.
.
CALL LINK("PARMGT",Oparameter$).
.
This subroutine retrieves a parameter left by the
PARMSV subroutine..
..
                      MISC.4.0.
..
.                      QUIT.
.
.
.
CALL LINK("QUIT").
.
This subroutine exits to the Master Title Screen. .
.
..
CAUTION: {Invoking this subroutine may destroy the
SXB} {subprograms depending on the particular version
of the} {console you are using.}.
.SP 44.
                      MISC.5.0.
..
.                     USERSUB.
.
.
.
CALL LINK("USERSUB"[,parameter-list]).
.
This subroutine is provided for those who would like
to use one of their own TMS9900 Assembly Language
subroutines in conjuction with SXB.  The following
```

```
rules must be strictly observed:.
.
1.  The program must begin with an absolute origin of
    >A000 (AORG >A000)..
.
2.  There must not be a DEF statement ... USERSUB has
    already been provided..
.
3.  The program must not be any longer than 256 bytes
    in length..
.
4.  The buffer areas starting at >A000+256 and
    A000+512 may be utilized for reading aand/or
    writing strings..
.
5.  If more than one subroutine is needed you can add
    a parameter to indicate which part of the
    subroutine to execute..
.
6.  J & K H SOFTWARE may at some future date provide
additional subroutines which would require the use of
this USESUB area.  They would be done in such a way so
as to be loaded on an optional basis so that you would
still be able to use your choice of subroutines (e.g.,
"yours" or "ours")..
..
                       MISC.6.0.
..

......................
```