Routine

COLLABORATORS			
	TITLE : Routine		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 3, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Routine

1.1 Routine Descriptions Guide

```
                              Routine.DESC file for DOOR.LIB

              Designed, compiled, and copyright 1993 by
                    Rick Rumer, aka The Technician



                 Distribution Notes

                 Recommendations

                 Library Functions

                 History of Changes to DOOR.LIB

                 Support and Assistance
```

1.2 distribution

```
     I  grant the right to use this code in any Tempest BBS application,
  for  whatever reason, to any individual willing to use it.  The author
  of Tempest BBS, Tim Hatzenbeler, may use any or all parts of this code
  as  he  sees  fit.   I grant permission for this to be released to the
  general  public  if he decides he wishes to.
     If  for  some  reason, Fred Fish decides that this code should be
released  to  the  public,  he is ALSO granted permission, even though
this Linktime-Library is Tempest BBS specific.
```

1.3 recommend

```
     Due to the size of the documentation, it is HIGHLY recommended that
  you print out the Routine.DESC file for off-line reference.  The pages
  have been designed to fit on a standard 60 line page, and should print
```

out fine on any printer. I have tried to make it one command per page, but several commands didn't fit, so you'll find the EXAMPLE code on the following page. I Do NOT recommend trying to print this .GUIDE file.

Note that pl() and CloseStuff() have been ALTERED from their original functions! They still perform the minimal duty they did, but now are more powerful!

1.4 techsupp

If you want commands added, or you find a bug, PLEASE call:

```
          ___________| __
         <___________|<  >=           NiteFall II BBS!
                     | \ \
                  /X\   > >  Featuring the BEST of the Public Domain!
               __/   \_/ /        (Amiga, MACintosh, IBM/Clones)
              / / ` ,') /
             /    \__/ /    2400 - 19.200 Baud     USR DUAL-STD MODEM!
            / /\      )           (300-1200 Message Bases Only)
    _______< <__)    /          (515) 277-1320   (Des Moines, IA)
   /        \ \     (
  /    \    <__>     \           SUPPORTING THE NEW MNP-5 MODEMS!
 //\/  /--\_____  __ |
// /  /  /     \ |  ||           654 Megs!  24 hrs, 7 days a week
\| \ |\ |       ||  ||
    || ||       ||  ||          Tempest BBS software support site!
    || ||       ||  ||
    |_\|_\      |_\ |_\          CALL NOW!!!! Access first call!
```

1.5 routines

AMPM

LockedOutText

CheckDoor

LOG

CheckKey

pl

CloseStuff

prompt

commas

```
Random

DEBUG

SetValue

DOORIO

ShowFile

FindUserSlot

TDHOTKEY

getkey

UserTime

GetStr

XmodemDownload

GetUserTime

yn

GetValue

ZmodemDownload

GetWorkDir

HitReturn

hotkey

input

LineInput

LoadSystemData
```

1.6 checkkey

```
NAME                                       USE : DOOR only

      int CheckKey()

SYNOPSIS

      #include <stdio.h>

      ret = CheckKey()

      int ret;       return code
      DESCRIPTION

   Check  to  see  if user hit a key while in a loop, and if they did,
set a flag.  (Does NOT wait for input, only checks to see if a key was
pressed  during  some  other  operation,  and then is queued up....)

                         Added  (VERSION 1.0)
      RETURNS

   A  return  value  of  0  indicates  all  went  well, and no key was
pressed.   A  return  of 1 indicates that there was indeed a character
pressed  (Such  as  an  abort  key).   This routine currently does NOT
return the actual key that was pressed.

      EXAMPLE

      #include <stdio.h>
      int IO;
      void main(void)
       {
        int x;
         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
            */
          {
           printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
              n");
           exit(0);
          }
        pl("OK.. In the door now....\r\n");
        for (x=0;x<900;x++)         /* A simple loop  counts 0->900 */
         {
          IO=CheckKey();            /* The actual checking          */
          if(!IO)                    /* if(IO!=0) do this, else go on*/
           {
            /* User pressed a key, so jump somewhere, or print
               something                                            */
            pl("You pressed a key!!! Aborting everything!\r\n");
            pl("Counter was at : %d\r\n",x); /* Note NEW pl() usage! */
            CloseStuff();     /* We aborted it all    */
           }
         }     /* else continue with loop */
        pl("You didn’t press any key, so I went on and on...\r\n");
        CloseStuff();
       }
```

1.7 doorio

```
      NAME                                          USE : DOOR only

            int  DOORIO();

      SYNOPSIS

            Not callable by users - STRICTLY for internal use!
      DESCRIPTION

   This  routine  serves  two  purposes.   First  it checks for a lost
carrier,  or  anything else that would signify that the door should be
abruptly  closed.   NO  actual closing is done here, but the EXIT_FLAG
variable is set appropriately (to a "1").
   Second,  it  processes  the  incoming/outgoing  "messages"  between
Tempest  BBS  and  your  DOOR  program.  This is the main link between
Tempest  and  your  DOOR.  Without this routine, you cannot use any of
the  commands  in  here.   This section will be automagically added to
every door program.

                         Added  (VERSION 1.0)

      RETURNS

   If this returns a 0, a "LOSS CARRIER" or similar event has occured,
and  the  EXIT_FLAG  is  set for the door to process.  Usually this is
done with :
#define  DROP     if(EXIT_FLAG) CloseStuff();

      SEE ALSO

   CheckDoor()
```

1.8 getvalue

```
      NAME                                       USE : DOOR only

            int  GetValue(int x);

      SYNOPSIS

            #include <stdio.h>

            status = GetValue(x)

            int status;    return value
            int x;         function to retrieve
                           0 = ANSI status

      DESCRIPTION

   This  function retrieves current values set in the BBS itself.  the
only current option is the User's current choice of ANSI, on or off.

                         Added  (VERSION 1.0)
      RETURNS

   A  0  indicates  the  user is NOT using ANSI, and any color or ANSI
positioning codes sent will be stripped out by the BBS.

      EXAMPLE
```

```
#include <stdio.h>
void main(void)
 {
  int AnsiColor;
  char ANSI[20];

   if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
      */
    {
     printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
        n");
     exit(0);
    }
   pl("OK.. In the door now....\r\n");
  AnsiColor=GetValue(0);
  if(AnsiColor)            /* If AnsiColor==0, it's off */
   {
    strcpy(ANSI,"OFF");
   }
  else
   {
    strcpy(ANSI,"ON");     /* Send'em all the ANSI you want */
   }
  pl("You have ANSI : %s\r\n",ANSI);
  CloseStuff();
 }

SEE ALSO
```

```
SetValue()
```

1.9 random

```
NAME                                        USE : DOOR only

      int  Random(int x);

SYNOPSIS

      #include <stdio.h>

      RndNum = Random(range);

      int RndNum;    /* A random number between 0 and x  */
      int range;     /* The highest possible number you want */

DESCRIPTION
```

```
   This  function  is  passed an integer value greater than 0, and the
routine  will  return  and  integer  value in the range of 0 to range.
This  is  a _seed generated random number, based on the VBLANK signal,
for the best possible random numbers.

                         Added  (VERSION 1.0)
```

```
      RETURNS

   A random integer value between 0 and range.

      EXAMPLE

      #include <stdio.h>
      void main(void)
       {
        int x=10;                  /* Select range: 0-10            */
        int RndNum;
         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
            */
          {
           printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
               n");
           exit(0);
          }
         pl("OK.. In the door now....\r\n");
        RndNum = Random(x);
        pl("Random number was : %d\r\n",RndNum);
        CloseStuff();
       }
```

1.10 checkdoor

```
      NAME                                       USE : DOOR only

            void CheckDoor(void);

      SYNOPSIS

            Not callable by users - STRICTLY for internal use!

      DESCRIPTION

   Checks for validity of a door, as a part of DoorStart();

                        Added  (VERSION 1.0)

      RETURNS

   None. (Alters pointers)
```

1.11 closestuff

```
      NAME                                       USE : DOOR only

            void CloseStuff(void);

      SYNOPSIS
```

```
            #include <stdio.h>

            CloseStuff();

      DESCRIPTION
```

Closes down the door. All open msgs that have not been replied to are taken care of, then the port deleted. After all housekeeping is finished, the door will exit with exit(0);
NOTE: This routine did NOT use to exit, but would let the door run. As of DOOR.LIB V1.0, this has been CHANGED, so that it WILL exit! This potentially avoids having lost tasks running.

```
                         Added  (VERSION 1.0)
      RETURNS
```

None.

```
      EXAMPLE

      #include <stdio.h>
      void main(void)
       {
         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
            */
          {
           printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
              n");
           exit(0);
          }
         pl("OK.. In the door now....\r\n");
        /* You would usually do something here.... */
        CloseStuff();      /* Close the door */
       }
```

1.12 getkey

```
      NAME                                       USE : DOOR only

            void getkey(char string[]);

      SYNOPSIS

            #include <stdio.h>

            getkey(char string[]);

            char string[255];   /*       The key pressed   */

      DESCRIPTION
```

This function is identical to hotkey(), with one exception. This routine does NOT filter out the cursor keys. This would be best used in a Full Screen Editor type function. (Which is what it was added for)

```
                        Added  (VERSION 1.0)

      RETURNS

   The key(s) pressed by the user. An input style function.

      EXAMPLE

      #include <stdio.h>
      void main(void)
       {
        char SomeChar[5], EndResult[255];
         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
            */
          {
           printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
               n");
           exit(0);
          }
         pl("OK.. In the door now....\r\n");
        pl("Please enter something.... Anything!\r\n");
        for(x=0;x<=100;x++)   /* Scan up to 100 times */
        {
         getkey(SomeChar);
         if(SomeChar[0]==13) break;  /* Aborted by pressing RETURN   */
         if(SomeChar[0]=='Y') break; /* Aborted by pressing "Y"      */
         if(SomeChar[0]=='N') break; /* Aborted by pressing "N"      */
         strcat(EndResult,SomeChar);
        }
        if(EndResult[0]='\0' || EndResult[0]=13) /* NULL or RETURN */
         {
          pl("You typed NOTHING!\r\n");
         }
        else
         {
          pl("You typed : %s\r\n",EndResult);
         }
        CloseStuff();      /* Close the door */
       }

      SEE ALSO

   hotkey(), TDHOTKEY(), input(), LineInput(), prompt()
```

1.13 getstr

```
      NAME                                       USE : DOOR only

            void GetStr(char string[],int opt);

      SYNOPSIS

            #include <stdio.h>

            GetStr(string,What)
```

```
     char string[255];      variable to hold the retrieved data
      int What;             What data to retrieve from the BBS
```

DESCRIPTION

Retrieves various data from the BBS structures located in RAM, depending on the value of the What variable.

Table of various values What can be :

```
 0 - The path the BBS was loaded with.
 1 - The path to the accounts data file.
 2 - The path to the catalog files are kept.
 3 - The path to the temporary directory.
 4 - The path to the Text directory.
 5 - The path to the Describe directory.
 6 - The path to the Voting Dir.
 7 - The path where the optional files are kept.
 8 - The path where the Modules are kept.
 9 - The path where the new user answers are kept.
10 - The path where sysop uploads are kept.
11 - The path where the aborted uploads are kept. (Resume dir)
12 - The path where uploads are kept when they're
     being uploaded. (work dir)
13 - The path where doors may be kept.
14 - The path where your log files are kept at.
15 - Get a full date & time string
16 - Get the current date
17 - Get the current time
18 - Get the system name of the bbs
19 - The Baud of the online caller.
```

Added (VERSION 1.0)

RETURNS

None. Variable Pointer string is altered.

EXAMPLE

```
#include <stdio.h>
void main(void)
 {
   int What=0;
  char string[255];

   if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
      */
    {
     printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
        n");
     exit(0);
    }
   pl("OK.. In the door now....\r\n");
  GetStr(string,What);  /* Get the BBS path, usually Tempest: or BBS: */
  pl("The BBS path is : %s\r\n",string);
```

```
         What+=1;              /* What = What + 1, ie now it's 1   */
         GetStr(string,what);
         pl("Your accounts.data file is at : %s\r\n",string);
         What+=1;              /* What = What + 1, ie now it's 2   */
         GetStr(string,what);
         pl("Your catalog files are in : %s\r\n",string);
         CloseStuff();      /* Close the door */
        }
```

1.14 getusertime

```
      NAME                                       USE : DOOR only

            void GetUserTime(char string[]);

      SYNOPSIS

            #include <stdio.h>

            void GetUserTime(char string[]);

            char string[255];   /* number of minutes left in "char" format */

      DESCRIPTION

   This  function  will  enable you to display the minutes that a user
has  remaining online in a string format.  To convert this to a number
for mathematical purposes, simply reference the example below.

                         Added (VERSION 1.0)

      RETURNS

   The minutes remaining online for the user.

      EXAMPLE

      #include <stdio.h>
      void main(void)
       {
        char string[255];
        int  HowMuch;
         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ←
            */
          {
           printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ←
               n");
           exit(0);
          }
         pl("OK.. In the door now....\r\n");
        GetUserTime(string);  /* Get the time left online          */
        pl("Warning! Only %s Minutes left online!\r\n",string);
        HowMuch=atoi(string);
        if(HowMuch < 15)
         {
```

```
         if(HowMuch < 10)
          {
          if(HowMuch < 5)
           {
           if(HowMuch <= 1)
            pl("EXIT IMMEDIATELY! YOU ARE OUT OF TIME!!\r\n");  goto Continue;
           }
          else pl("FIVE MINUTE WARNING!\r\n");  goto Continue;
          }
         else  pl("TEN MINUTE WARNING!\r\n");  goto Continue;
         }
        else   pl("FIFTEEN MINUTE WARNING!\r\n");  goto Continue;

      pl("You've got plenty of time left! Have fun!\r\n");
      Continue:
      /* Your other stuff here.... then check again, etc... */
      CloseStuff();
      }

      SEE ALSO

   UserTime()
```

1.15 getworkdir

```
      NAME                                       USE : DOOR only

            void GetWorkDir(char string[]);

      SYNOPSIS

            #include <stdio.h>

            void GetWorkDir(char string[]);

            char string[255];   /* Path to #?.Data files    */

      DESCRIPTION

   This  command  tells  you where the #?.data files can be found.  If
you use the normal setup, this will be "Tempest:Setup/".  This command
is  here  because it is can be somewhere else if the SYSOP chose to do
that. The files located in that directory are currently :

   BAUD.data                  Keys.data
   Bulletins.data             MESSAGES.data
   Color.data                 Modem.data
   CONFIG.data                PRESETS.data
   Custom.data                Protocol.data
   Doors.data                 Questions.data
   ExtraPaths.data            Reserved.data
   FILES.data                 RunTime.data
   Internal.data

                         Added (VERSION 1.0)
      RETURNS

   None. Alters pointer to show complete path to the #?.Data files.

      EXAMPLE

      #include <stdio.h>
      void main(void)
       {
        char string[255];

         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
            */
          {
           printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
               n");
           exit(0);
          }
         pl("OK.. In the door now....\r\n");
        GetWorkDir(string);
        pl("The path to your #?.Data files is : %s\r\n",string);
        CloseStuff();      /* Close the door */
       }
```

1.16 hotkey

```
      NAME                                       USE : DOOR only

            void hotkey(char string[]);

      SYNOPSIS

            #include <stdio.h>

            void hotkey(char string[]);

            char string[5];   /* Key Pressed by user */

      DESCRIPTION

   This waits until 1 key has been pressed.  It does not count control
characters, nor cursor keys.

                         Added (VERSION 1.0)

      RETURNS

   None. Alters pointer to contain the key pressed. (RETURN=13 or '\n')

      EXAMPLE

      #include <stdio.h>
      void main(void)
       {
        char string[255];

         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
            */
          {
           printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
              n");
           exit(0);
          }
         pl("OK.. In the door now....\r\n");
        hotkey(string);       /* Wait for an inputted character   */
        pl("The key you pressed was : %s\r\n",string);
        CloseStuff();      /* Close the door */
       }

      SEE ALSO

   TDHOTKEY(), input(), LineInput(), getkey()
```

1.17 input

```
      NAME                                       USE : DOOR only

            void input(char string[],int len);

      SYNOPSIS

            #include <stdio.h>

            void input(char string[],int len);

             int len;           /* Number of allowed characters   */
            char string[255];   /* The string entered             */

      DESCRIPTION

   This command accepts input from the user until either the user hits
return,  or  they  enter the maximum number of characters allowed.  if
they  reach  the  maximum,  it  stops accepting input, and waits for a
RETURN press.

                        Added (VERSION 1.0)

      RETURNS

   None. Alters pointer to show user's input.

      EXAMPLE

      #include <stdio.h>
      void main(void)
       {
        char string[255];
```

```
         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
            */
          {
           printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
               n");
           exit(0);
          }
         pl("OK.. In the door now....\r\n");
        pl("You have 15 characters to describe yourself in.\r\n");
        input(string,15);
        pl("You typed : %s\r\n",string);
        CloseStuff();      /* Close the door */
       }
```

1.18 lineinput

```
      NAME                                       USE : DOOR only

            void LineInput(char string1[],char string2[],int len);

      SYNOPSIS

            #include <stdio.h>

            void LineInput(char string1[],char string2[],int len);

             int len;           /* Number of allowed characters   */
            char string1[255];   /* The original string entered   */
            char string2[255];   /* The new string entered        */

      DESCRIPTION

   This  command  is identical to the input() command, except that the
user  is allowed to edit a default, or perhaps a previous string.  The
string1 will be printed first, then the cursor is placed at the end of
the  string  for input/editing.  The completed and edited string is in
string2.

                         Added (VERSION 1.0)

      RETURNS

   None. Alters pointer to show user's input.

      EXAMPLE

      #include <stdio.h>
      void main(void)
       {
        char string1[255],string2[255];

         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
            */
          {
```

```
          printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
             n");
          exit(0);
         }
       pl("OK.. In the door now....\r\n");
      pl("You have 15 characters to describe yourself in.\r\n");
      input(string1,15);
      pl("You typed : %s\r\n",string1);
      pl("I'll let you edit it in case you made a mistake.\r\n");
      input(string1,string2,15);
      pl("NOW the string is %s\r\n",string2);
      CloseStuff();      /* Close the door */
     }
```

1.19 loadsystemdata

```
    NAME                                       USE : DOOR only

          void LoadSystemData(void);

    SYNOPSIS

          #include <stdio.h>

          void LoadSystemData(void);

    DESCRIPTION

   This is identical to the SYSOP hitting F7 when the bbs is idle.  It
will  load up all the #?.data files into the BBS structures, Using any
new settings that may have been altered since the BBS was run.

                         Added (VERSION 1.0)

    RETURNS

   None.

    EXAMPLE

    #include <stdio.h>
    #include <DOS.h>

    void main(int argc, char *argv[]);
     {
       if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
          */
         {
          printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
             n");
          exit(0);
         }
       pl("OK.. In the door now....\r\n");
      DeleteFile("BBS:Setup/Message.DATA");
      /* redefine message areas here */
```

```
        LoadSystemData();
        pl("Message base changed, as you requested!\r\n");
        CloseStuff();      /* Close the door */
       }
```

1.20 pl

```
      NAME                                      USE : DOOR only

            void pl(char string[]);

      SYNOPSIS

            #include <stdio.h>

            void pl(char *format, arg1, arg2...arg8);

            char *format;   /* The formatted string to send      */

      DESCRIPTION
```

This command has been rewritten to allow new paramters. Don't let the command line fool you, it's not hard to use! It's also 100% compatible with any door written for Tempest. Just now has more options. If you are familiar with the printf() function, this now operates the SAME. You may use UP TO 8 arguments, anything after that will be ignored. (The same limits apply to printf() and sprintf() as well, so there shouldn't be any problems with this.)

Added (VERSION 1.0)

```
      RETURNS
```

None.

```
      EXAMPLE

      #include <stdio.h>
      void main(int argc, char *argv[])
       {
        char string[255];
         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
            */
          {
           printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
              n");
           exit(0);
          }
         pl("OK.. In the door now....\r\n");
                  /* Example of NO formatting   */
        pl("You have 15 characters to describe yourself in.\r\n");
        input(string,15);
                  /* Example of formatting with 1 argument  */
        pl("You said : %s\r\n",string);
        CloseStuff();      /* Close the door */
       }
```

1.21 setvalue

```
      NAME                                       USE : DOOR only

            void SetValue(int x,int y);

      SYNOPSIS

            #include <stdio.h>

            void SetValue(int x,int y);

            int x;      /* Value             */
            int y;      /* Command           */

      DESCRIPTION

   This  allows you to alter the BBS priority, or turn ANSI on or off.
You  may  want  to  turn ANSI on for cursor positioning codes, or some
other reason.

                         Added (VERSION 1.0)
      EXAMPLE

      #include <stdio.h>
      void main(int argc, char *argv[])
       {
         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
            */
          {
           printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
              n");
           exit(0);
          }
        SetValue(1,-5);    /* Set BBS priority to -5           */
        pl("BBS Priority changed to -5.\r\n");
        SetValue(0,1);    /* Set ANSI to ON                   */
        pl("?[36mA?[34mN?[33mS?[32mI ?[35mis ?[31mnow ?[37mON?[0m\r\n");
        SetValue(0,0);    /* Set ANSI to OFF                  */
        pl("ANSI is now OFF\r\n");
        CloseStuff();      /* Close the door , and return the BBS
                              priority back to normal, automatically */
       }

      SEE ALSO

   GetValue()
```

1.22 showfile

```
      NAME                                       USE : DOOR only

            void ShowFile(char string[]);

      SYNOPSIS

            #include <stdio.h>

            void ShowFile(char string[]);

            char string[255];       /* Path & Filename to view */

      DESCRIPTION

   The  file  will be shown (if found) and ANSI will be displayed, and
any  "~"  commands  within that file will be executed as normal.  This
DOES  include other doors, etc.  HOWEVER, running a DOOR _FROM_ a DOOR
_WILL_  crash the system!  The message ports are designed for only one
door to be open at a time per Tempest Node!

                         Added (VERSION 1.0)

      EXAMPLE

      #include <stdio.h>
      void main(int argc, char *argv[])
       {
        char Filename[255];
         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
             */
          {
           printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
               n");
           exit(0);
          }
        strcpy(Filename,"BBS:text/logoff.txt");
        ShowFile(Filename);      /* Display the logoff.txt file to user */
        CloseStuff();
       }
```

1.23 usertime

```
      NAME                                       USE : DOOR only

            void UserTime(int x);

      SYNOPSIS

            #include <stdio.h>

            void UserTime(int x);

            int x;      /* the amount of time to add/subtract  */
      DESCRIPTION

   Either  adds  or subtracts online time for the user.  If the number
(x) is negative, it will subtract time, if positive, it will add time.

                         Added (VERSION 1.0)

      EXAMPLE

      #include <stdio.h>
      void main(int argc, char *argv[])
       {
        int Time;

        Time=15;
         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
            */
          {
           printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
               n");
           exit(0);
          }
        UserTime(Time);      /* Adds 15 minutes to their current time */
        Time = Time * -1;    /* Convert to a negative number          */
        UserTime(Time);      /* Subtracts 15 minutes from their time  */
        CloseStuff();
       }

      SEE ALSO

   GetUserTime()
```

1.24 xmodemdownload

```
      NAME                                       USE : DOOR only

            void XmodemDownload(char string[]);

      SYNOPSIS

            #include <stdio.h>

            void XmodemDownload(char string[]);

            char string[255];    /* Filename to d/l            */

      DESCRIPTION

   This  allows  users  to  d/l  something  your door created, or made
available to them.  Obviously this function uses X-Modem protocol, and
it  can  only  do  one file at a time.  This is here for compatibility
only,  and  if  at  ALL  possible,  your  door  should be using Zmodem
instead!  This  protocol  is  the  SLOWEST available!  If called from
LOCAL/VIEW  mode,  this  command will fail, and quietly abort, without
error or complications.
```

```
                    Added (VERSION 1.0)

     EXAMPLE

     #include <stdio.h>
     void main(int argc, char *argv[])
      {                    /* d/l my Startup-sequence             */
        if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
           */
         {
          printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
             n");
          exit(0);
         }
       XmodemDownload("S:Startup-Sequence");
       CloseStuff();
      }

     SEE ALSO

   ZmodemDownload()
```

1.25 zmodemdownload

```
     NAME                                      USE : DOOR only

           void ZmodemDownload(char string[]);

     SYNOPSIS

           #include <stdio.h>

           void ZmodemDownload(char string[]);

           char string[255];    /* Filename to d/l            */

     DESCRIPTION

   This  allows  users  to  d/l  something  your door created, or made
available  to them.  This protocol is the FASTEST currently available!
If  called  from  LOCAL/VIEW mode, this command will fail, and quietly
abort,  without error or complications.  This protocol can ALSO handle
batch sending, should the need arise.

                    Added (VERSION 1.0)

     EXAMPLE

     #include <stdio.h>
     void main(int argc, char *argv[])
      {                    /* d/l my Startup-sequence             */
        if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
           */
         {
          printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
              n");
          exit(0);
         }
       ZmodemDownload("S:Startup-Sequence");   /* Example of multiple files */
       ZmodemDownload("S:User-Startup");
       ZmodemDownload("S:Shell-Startup");
       CloseStuff();
      }

      SEE ALSO

   XmodemDownload
```

1.26 tdhotkey

```
      NAME                                       USE : DOOR only

            int TDHOTKEY(char str[], char des[])

      SYNOPSIS

            #include <stdio.h>

            stat = TDHOTKEY(char str[], char des[])

             int stat;        /* Dummy variable, always 0   */
            char str[255];    /* string to print            */
            char des[255];    /* string to record answer in */

      DESCRIPTION

   This  is  identical  to Tim's hotkey() routine, with ONE exception.
In  addition to getting a character from the keyboard, it also outputs
a string first.  That way you don't have to do a pl(string) and then a
hotkey(string)  kinda thing.  Makes it more compact.  This routine was
NOT part of the initial package, and was added by me (The Technician).

                         Added (VERSION 1.0)
      EXAMPLE

      #include <stdio.h>
      void main(int argc, char *argv[])
       {
        char string[255];
                             /* Wait for an inputted character   */
         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
             */
          {
           printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
               n");
           exit(0);
          }
        TDHOTKEY("What's it gonna be? [Y/n] ",string);
        pl("The key you pressed was : %s\r\n",string);
```

```
         CloseStuff();      /* Close the door */
        }

      SEE ALSO

   getkey(), hotkey(), input(), LineInput()
```

1.27 prompt

```
      NAME                                      USE : DOOR only

            int prompt(char str[], char des[], int len)

      SYNOPSIS

            #include <stdio.h>

            int prompt(char str[], char des[], int len)

            char str[255];    /* string to print               */
            char des[255];    /* string to get                 */
             int len;         /* Maximum length allowed        */

      DESCRIPTION

   This  is  the  same as input, but displays a string first.  You are
NOT  allowed  to  edit  the  string, as it id for information purposes
only!

                         Added (VERSION 1.0)

      EXAMPLE

      #include <stdio.h>
      void main(int argc, char *argv[])
       {
        char print[255];
        char input[255];
         int len=0;
         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
            */
          {
           printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
               n");
           exit(0);
          }
        strcpy(print,"Input something, please");
        len=35;
        prompt(print, input, len);
        pl("you entered, %s\r\n",input);
        CloseStuff();
       }

      SEE ALSO
   getkey(), input(), LineInput(), HOTKEY, hotkey()
```

1.28 hitreturn

```
      NAME                                       USE : DOOR only

            void HitReturn(int CR_LF);

      SYNOPSIS

            #include <stdio.h>

            void HitReturn(int CR_LF);

            int CR_LF;        /* Number of \r\n to send        */

      DESCRIPTION

   This  is  a  quick  way  to display some information (Prior to this
call)   and   have   the   BBS   wait   for  them  to  read  it.   The
carriage-returns/linefeeds  ("\r\n")  will  be  sent BEFORE the "Press
[RETURN]"  prompt,  so  you  can  correctly space out your data.  This
routine does not return any values.

                         Added (VERSION 1.1)

      EXAMPLE

      #include <stdio.h>
      void main(int argc, char *argv[])
       {
         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
            */
          {
           printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
              n");
           exit(0);
          }
        pl("Ya can’t do that, stupid user!");
        HitReturn(2);
        CloseStuff();
       }

      OUTPUT

      Ya Can’t do that, stupid user!


      Press [RETURN]
```

1.29 commas

```
      NAME                                        USE : DOOR or CLI

             char *commas(long number, char buffer[])

      SYNOPSIS

            #include <stdio.h>

            char *commas(long number, char buffer[])

            long number;       /* Number to insert commas in  */
            char buffer[];     /* buffer pointer              */

      DESCRIPTION

   This  small  routine ( [C] Jabba Development ) takes a long number,
and  inserts  ","  (Comma) characters in every third place, so to make
the number much more readable.  buffer[] is the return string pointer,
and will contain, after the call, the long number with commas properly
inserted.   Please  note!  1) The char buff[] used below is defined in
the  door code itself, and MUST be declared in YOUR program as "extern
char  buff[255]"  !!   2)  You  can  ONLY  have ONE call to the commas
routine PER pl statement!

   This is illegal:

pl("UL : %s   DL : %s\r\n", commas(ulbytes,buff), commas(dlbytes,buff) );

   The CORRECT way to do the above:

pl("UL : %s   ",  commas(ulbytes,buff) );   /* (NOTICE _NO_ \r\n!) */
pl("DL : %s\r\n", commas(dlbytes,buff) );

                         Added (VERSION 1.2)

      EXAMPLE

      #include <stdio.h>
      void main(int argc, char *argv[])
       {
        long number=12345678L;
        extern char buff[255];

         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
            */
          {
           printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
               n");
           exit(0);
          }
        pl("Before: number=%ld \r\n",number);
        pl("After : number=%s \r\n",commas(number,buff) );
        CloseStuff();
       }
```

1.30 debug

```
NAME                                       USE : DOOR only

      void DEBUG(fmt,a1,a2,a3,a4,a5,a6,a7,a8)

SYNOPSIS

      #include <stdio.h>

      void DEBUG(fmt,a1,a2,a3,a4,a5,a6,a7,a8)

      char *fmt;        /* Formatting parameters      */

DESCRIPTION
```

This is similar to the pl() function, except thet instead of writing to the screen, this writes to the BBS:LOGS/DE_BUG.LOG. It automatically reads YOUR path, so it is NOT hardcoded to the above path. (Consider it an example...) This is useful if you're having a bug in a door that is hard to track down... You can still let the users use the door, and print out variable values, etc, to the DE_BUG.LOG, for later in-depth examination. In this manner you can test doors as you write them. The format is identical to theat of pl() and printf(). The same limit applies though, no more than 8 variables per DEBUG() call!

Added (VERSION 1.2)

```
EXAMPLE

#include <stdio.h>
void main(int argc, char *argv[]);  /* Prorotype of main() */
extern int EXIT_FLAG;      /* For below */
#define DROP if(EXIT_FLAG) CloseStuff();  /* For the DROP routine */

 void main(int argc, char *argv[])
  {
   if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
      */
    {
     printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
        n");
     exit(0);
    }
   pl("OK.. In the door now....\r\n");
   DEBUG("This is a test of the emergency broadcast ");
   DEBUG("System. \r\n If this had been a real emergency,");
   DEBUG("Then I would have crashed!\r\n");
   pl("Wrote to the file, and am exiting now!\r\n\r\n");
   CloseStuff();        // Now contains the exit routine.....
  }

SEE ALSO
```

pl(), LOG()

1.31 log

```
NAME                                       USE : DOOR only

      void LOG(fmt,a1,a2,a3,a4,a5,a6,a7,a8)

SYNOPSIS

      #include <stdio.h>

      void LOG(fmt,a1,a2,a3,a4,a5,a6,a7,a8)

      char *fmt;        /* Formatting parameters      */

DESCRIPTION
```

This is similar to the pl() function, except thet instead of writing to the screen, this writes to the BBS:LOGS/.LOG. It automatically reads YOUR path, so it is NOT hardcoded to the above path. (Consider it an example...) This is IDENTICAL to the DEBUG() routine, except for the file it writes to. The format is identical to theat of pl() and printf(). The same limit applies though, no more than 8 variables per LOG() call!

Added (VERSION 1.2)

```
EXAMPLE

#include <stdio.h>
void main(int argc, char *argv[]);  /* Prorotype of main() */
extern int EXIT_FLAG;      /* For below */
#define DROP if(EXIT_FLAG) CloseStuff();  /* For the DROP routine */

 void main(int argc, char *argv[])
  {
   if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
      */
    {
     printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
        n");
     exit(0);
    }
   pl("OK.. In the door now....\r\n");
   LOG("I decided to write this info into\r\n");
   LOG("The SYSOP's log, just to record it permanently, and to\r\n");
   LOG("Catch his/her full attention......\r\n");
   pl("Wrote to the file, and am exiting now!\r\n\r\n");
   CloseStuff();        // Now contains the exit routine.....
  }

SEE ALSO
```

pl(), DEBUG()

1.32 yn

```
      NAME                                       USE : DOOR only

            void yn(fmt,a1,a2,a3,a4,a5,a6,a7,a8)

      SYNOPSIS

            #include <stdio.h>

            void yn(fmt,a1,a2,a3,a4,a5,a6,a7,a8)

            char *fmt;        /* Formatting parameters      */

      DESCRIPTION

   This  is  variation  on a few routines.  It will print out a string
first  (Optional, as it checks for a null string), then requires a "Y"
or "N" input.  A Cariage return is NOT acceptable.  It uses the hotkey
command,  so  hitting return is not neccessary.  It accepts both upper
and lowercase.

                          Added (VERSION 1.3)

      EXAMPLE

      #include <stdio.h>
      void main(int argc, char *argv[]);  /* Prototype of main() */
      extern int EXIT_FLAG;      /* For below */
      #define DROP if(EXIT_FLAG) CloseStuff();  /* For the DROP routine */

       void main(int argc, char *argv[])
        {
         int uhg=16;
         long erp=123456789;
         char stupid[255];
         strcpy(stupid,"Stupid text for testing!");

         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
            */
          {
           printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
               n");
           exit(0);
          }
         pl("OK.. In the door now....\r\n");

         if(yn("uhg=%d, erp=%ld, and stupid is %s\r\nDo you like cottage cheese ?  ↩
            [Y/N] ",uhg,erp,stupid) )
            pl("Yeah? I like it too!\r\n");  /* They answered YES, YN() returned a ↩
                1*/
         else
            pl("I don't blame you! It looks funny!\r\n");   /* Answered NO! YN()  ↩
               returned a 0   */
         HitReturn(2);
         pl("exiting now!\r\n\r\n");
```

```
     CloseStuff();        // Now contains the exit routine.....
    }
```

1.33 ampm

```
NAME                                       USE : DOOR or CLI

      void AMPM(char time[255], char ampm[5])

SYNOPSIS

      #include <stdio.h>

      void AMPM(char time[255], char ampm[5])

      char time[255];        /* The time to convert (23:13) */
      char ampm[5];          /* string that contains " am" or " pm" */

DESCRIPTION
```

```
   This  takes  a  time  string,  and converts it from military tim to
normal  time.   For  instructions  on how to get this time from a LONG
value, see the example below.  Both time AND ampm will be modified, so
if  you  need to use the original data later in your program, then you
must  save  it  off before calling the routine.  An example of this is
also seen in the below example.

                         Added (VERSION 1.4)
```

```
EXAMPLE

#include <stdio.h>
void main(int argc, char *argv[]);  /* Prototype of main() */
extern int EXIT_FLAG;      /* For below */
#define DROP if(EXIT_FLAG) CloseStuff();  /* For the DROP routine */

 void main(int argc, char *argv[])
  {
   char string[255],
        backup[255],
            ampm[5];
   /* This is the format for the CURRENT time on YOUR Amiga    */
   GetStr(string,17);   /* Get AMIGA time */
   /* This would be the format for getting the time a user was last on! */
   //   strcpy(string,ctime(&Last[x].Time_Last));
   //   strmid(string,time,12,5);

   if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
      */
    {
     printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
        n");
     exit(0);
    }
   pl("OK.. In the door now....\r\n");
```

```
         strcpy(backup,string);  /* Make a copy of the data for later use */
         AMPM(string,ampm);
         pl("before: %s, after: %s%s\r\n",backup,string,ampm);

         HitReturn(2);
         pl("exiting now!\r\n\r\n");
         CloseStuff();        // Now contains the exit routine.....
        }
```

1.34 finduserslot

```
      NAME                                       USE : DOOR only

            int FindUserSlot(char Handle[])

      SYNOPSIS

            #include <stdio.h>

            int FindUserSlot(char Handle[])

            char Handle[255];      /* Users handle to search for  */

      DESCRIPTION

   This  function  serves  two  purposes.   The first thing it does is
search  the  Accounts.INX  file  for  a  specific users name, which is
passed  to it through the Handle[] variable.  The second, is to verify
that a user exists.  A return code of -1 means either the Accounts.INX
file  couldn't  be found, or that it couldn't be opened.  A returncode
of  0  indicates  the handle/user doesn't exist.  Any other returncode
ABOVE 0 indicates that the user DOES exist, and the number returned is
their  slot  number.   If  you  wish  to use the number returned for a
Seek() command, then you must subtract 1 from the returned value.

                         Added (VERSION 1.5)

      EXAMPLE

      #include <stdio.h>
      void main(int argc, char *argv[]);  /* Prototype of main() */
      extern int EXIT_FLAG;      /* For below */
      #define DROP if(EXIT_FLAG) CloseStuff();  /* For the DROP routine */

       void main(int argc, char *argv[])
        {
         char name[255];
         int  hmm=0;
         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
            */
          {
           printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
               n");
           exit(0);
          }
```

```
     pl("OK.. In the door now....\r\n");
     strcpy(name,"The Technician");         /* Let's look for ME */
     hmm=FindUserSlot(name);
     if(hmm==-1) pl("Your Accounts.INX file is missing or corrupt!\r\n");
     if(hmm==0) pl("I've never called your BBS!\r\n");
     if(hmm>0) pl("I am User # %d on your BBS!\r\n",hmm);
     HitReturn(2);
     pl("exiting now!\r\n\r\n");
     CloseStuff();        // Now contains the exit routine.....
    }
```

1.35 lockedouttext

```
      NAME                                       USE : DOOR only

            LockedOutText(char Text[], int Which)

      SYNOPSIS

            #include <stdio.h>

            LockedOutText(char Text[], int Which)

            char Text[255];      /* Name/Password to check    */
            int  Which;          /* Do name or password check */

      DESCRIPTION

   This  function  serves  two purposes.  To pick which option to use,
you  must  define  the Which variable before the call.  If you want to
search  the  NAMES.OPT file, pass a "1" to the Which variable.  If you
wish to search the PASSWORDS.OPT file, pass it a "2".  In either case,
the return values are as follows:

                -1 = Error opening the file
                 0 = No match found
                 1 = Found a match

                         Added (VERSION 1.5)

      EXAMPLE

      #include <stdio.h>
      void main(int argc, char *argv[]);  /* Prototype of main() */
      extern int EXIT_FLAG;      /* For below */
      #define DROP if(EXIT_FLAG) CloseStuff();  /* For the DROP routine */

       void main(int argc, char *argv[])
        {
         char text[255];
         int  hmm=0;
         if(!DoorStart(argv[1]) || argc <2 ) /* REQUIRED! Door locks without it!  ↩
             */
          {
```

```
      printf("Sorry. I am a DOOR program for Tempest BBS, NOT an executable!\ ↩
          n");
      exit(0);
     }
    pl("OK.. In the door now....\r\n");
    strcpy(text,"The Technician");
    hmm=LockedOutText(text,1);
    if(hmm==-1) pl("Your NAMES.OPT file is missing or corrupt!\r\n");
    if(hmm==0) pl("I'm not locked out of your BBS!\r\n");
    if(hmm==1) pl("Uh oh! I am Locked out! Aaaack!\r\n");

    strcpy(text,"PASSWORD");
    hmm=LockedOutText(text,2);
    if(hmm==-1) pl("Your PASSWORDS.OPT file is missing or corrupt!\r\n");
    if(hmm==0) pl("the password PASSWORD isn't locked out\r\n");
    if(hmm==1) pl("PASSWORD cannot be used on this BBS!\r\n");
    HitReturn(2);
    pl("exiting now!\r\n\r\n");
    CloseStuff();        // Now contains the exit routine.....
   }
```

1.36 changes

```
This section is organized from the most recent changes to the oldest.

                     DOOR.LIB V1.5 ADDITIONS:

   New  to  this version are new USE parameters.  Some of the routines
used in here do not make actual calls to DOOR functions, so they could
be  used  in  CLI  programs/utilities  as well.  Since only I have the
sourcecode, and you wouldn't know for sure, I have made up a new field
in this document, called USE.  In this area I will define if it can be
used from the CLI, as a DOOR, or both.
   Added  the  LockedOutText() and FindUserSlot() routines.  These are
pretty   self   descriptive,   so   I   won't  go  into  detail  here.
LockedOutText()  deserves  a  slight  explanation.   It can search the
NAMES.OPT  _OR_  the  PASSWORDS.OPT  file.   These  are  VERY powerful
commands,  and  we  can  thank  "The Skeleton" for these quick & dirty
hacks!  They are fast, and useful to almost everyone, myself included!
   I  also  edited  EVERY  commandin  this  document,  making sure the
example  code was perfect.  I discovered upon reading through it, that
most  of  the  code  was  incomplete.   As  of  now, ALL code given as
examples  is  100%  compatible  and  compilable  as a stand alone DOOR
program.   (Given that you're using SAS/C 5.1b or better!) If you find
an example that isn't so, please let me know so it can be corrected...

                     DOOR.LIB V1.4 ADDITIONS:

   Added  The Skeleton's AMPM() military time converter routine.  This
version was NOT released.

                     DOOR.LIB V1.3 ADDITIONS:

   Added the YN() routine. This version was NOT released !
```

DOOR.LIB V1.2 ADDITIONS:

Added routine Commas(). This really WAS included in 1.0, but I forgot to document it’s use. SORRY!
Added routine DEBUG() for use with debugging door code you write!
Added routine LOG() for writing to the Sysop’s LOG file. You can place warnings, error results, etc here, so the sysop is sure to see it.

DOOR.LIB V1.1 ADDITIONS:

Added routine "HitReturn(CR_LF)" for printing CR_LF number of carriage returns/linefeeds, then following it with a "Press RETURN" style prompt. I use this in 99% of my personal code, so it was added here for the convenience of others as well.