O V E R L I N K   L I B R A R Y   # 1
DarkStar CBBS Modular Expansion ToolKit

The New Era of Open Architecture BBSing

Our warranty policy follows that outlined on the first page of the BBS V3.0 manual. Any defective media may be returned at a replacement cost of $7.50 (U.S. funds) for either the OverLink disk or the V3.1 upgrade disk, or $15.00 (U.S. funds) for both disks, provided that you follow the guidelines of the warranty policy outlined in the V3.0 manual.

The OverLink - Contents

PUBLISHER'S NOTE

We Hope You Enjoy This First Library/Expansion Disk. We Are Breaking New Ground With Our Unique Approach! We Invite All Comments andSuggestions Relating to Ideas That You Would Like to See Implemented in Future Library Editions! If You Can Provide Us With a Basic Version of Your Idea, We Will Make Every Reasonable Effort to Convert It to ML For the BBS!

If You Would Like to Become a DarkStar Distributor in Your Area, then By All Means, Drop Us a Line and We Will Provide You With Further Details.

Andrew Leaver
President, D.S.S.

DarkStar BBS V3.1

Version 3.1 is a slightly revised version of 3.0, with only a few minor changes being made, along with one major change. Although the manual stated that version 3.0 would work with the GameLink expansion system, this is not the case anymore. The original GameLink design was dropped due to problems with the basic routine calls in the V3.0 program. Major changes were made internally to V3.1 to accomodate the new OverLink, and also the new CWI. A full explanation of the OverLink system is given later.

There was one minor bug that was corrected to the system with respect to forwarding messages on a dual drive. If you had placed your message text on drive 1 of a dual drive, you would find that forwarding messages did not rename the text file. This has been corrected.

The basic screen editor system of DarkTerm/DarkStar BBS has been modified to allow the cursor to be plotted on the screen, so that it will be able to "jump" from one (row,column) position to another. On the BBS V3.1 disk #1, you will find a file called DARKTERM 4.C. This file is a boot that installs the plot routine on DarkTerm 4.B. To use it, just type LOAD"darkterm 4.c",8,1. When you see the status line on the terminal now, it will indicate that you are now using version 4.c. NOTE: you cannot use the EDITLINK with the 4.c boot. For this reason, do not get rid of your V4.b boot. You must use V4.b to use the edit link. The plot routine is in effect automatically when using the BBS.

The plot function operates as follows: enter CTRL-V. This puts the print routine into plot mode. Follow CTRL-V by entering a 2 digit number for the column for the cursor to jump to. Follow this with a 2 digit number for the new row for the cursor. You can see that this makes up a 5 character string. It MUST be 5 characters in length. Thus, if you want to put the cursor at column 1, you must enter a "01", not "1". For example, to move the cursor to column 20, row 10, enter "CTRL-V2010", where CTRL-V is the key you press with control and v at the same time. To move the cursor to position 1,1, you would enter CTRL-V0101.

Cursor plotting will be used in future on-line games for the CBBS, so become familiar with it; you'll need it. If a terminal other than DarkTerm 4.c is used, then the plot will not go through. Instead, the user would only see the number strings. The plot routine ensures that improper plot calls will not work. For example, CTRL-V9856 would not move the cursor to position 98,56. Remember that the cursor tracker in the BBS and terminal status lines displays the cursor position in row,column format, whereas the plot routine requires a column,row input.

V3.1 uses a different copy protection scheme than V3.0 uses. The new method requires that your disk drive that you boot the BBS from be in proper alignment. This protection type is not one of those infamous "head-banging" schemes, so relax. If the program does not load, then you should a) lift the drive door, remove the disk, put it back in, then close the door again (the head will not bang if you do this), b) have your drive aligned, or c) don't use a backup copy, which won't work at all (at this point in time). Only disk #1 is copy protected. Disk #2 is the same disk used with the V3.0 system. Read the BBS V3.0 manual for information regarding that disk. The program will load off most 1541 compatible drives, not just the true blue 1541.

If you look carefully at the other manual you have, you will see that it is a V3.0 manual. BBS V3.1 is EXACTLY the same as V3.0, EXECPT for the few changes which are mentioned in this manual. The BBS structure has not changed, nor have the commands and their functions.

NOTE: You must not have a printer connected to your computer, even if it is not turned on. Make sure you do not have anything plugged into the cassette port on the computer. When you boot the BBS up, be sure to turn the computer off then on, so that the memory is reset to its power-on state.

A second change to the BBS was made to allow the OverLink system to better communicate with the BBS. Both the E (Edit File) and $ (Disk Commands) command in the editor section access the disk by selecting a system drive to operate with. The V3.0 BBS used F,M,B and the 8 physical directories (1-8) to specify a drive. With OverLink, the BBS can operate on many more drives, which may not be set to one of the 11 drives above. To remedy this, V3.1 uses a drive table that is created with INSTALL DRIVES, which can be found on your OverLink disk. This small BASIC program installs up to 32 drives on the system that can be accessed by E,R and $ by entering the 2 digit number of the drive you want to access. For example, when you get the "$>" prompt with disk commands, you would enter "01" to access drive 1 in the drive table. Remember to use 2 digits! Entering "1" will only give you a disk error #31. If you attempt to access drive 56 by entering "56", this will cause the BBS to set the current drive to the device of drive #2 in the drive table. This is to prevent illegal drive table access.

Setup procedures for V3.1 basically follow those of V3.0. Working with OverLink is a separate task altogether. That will be explained later. Just note that you should perform both the BBS AND OverLink setup procedures before you run the BBS. The order is not relavent.

Setup procedures for V3.1 are outlined below.

1. Create a customized version of the BBS setup using the utilities on disk #2. This is outlined in the V3.0 manual. As a suggestion, take the files the BBS needs to run with from disk #2, and place them on a new disk. This will make working with the BBS a little faster and easier. The files the BBS needs for execution are :

CONFIGURATION	: The BBS configuration created with CREATE.CNF
CATEGORIES	: The message categories file created with CREATE.CAT
?.COMMANDS	: The BBS commands file. The value of "?" will be a C for color, or an A for ascii, created with CREATE.COM
COLOR TABLE	: The CBBS command color table, created with CREATE.COL.
SYSOP NAME	: Sysop name file, created with CREATE.NAM.
MODEM FILE	: The modem file you want to use (ie., MODEM.1650 for 1650's).
TEXT FILE	: The BBS text file. This defaults to BBS.TXT, but can be changed to any name and style with CREATE.TXT.
TIMELOK	: The restricted time period file, created with CREATE.TIM.
DRIVE TABLE	: The V3.1 drive table, created with INSTALL DRIVES.

All of the above are set up with the V3.0 system, except for DRIVE TABLE. See the V3.0 manual for setup procedures. Copy INSTALL DRIVES to your disk #2 backup, so that it will reside with the other setup files.

INSTALL DRIVES. This very short BASIC program sets up the drive table. Load it, and change the data statements in lines 1060-1090 and lines 1130-1160. There is data for 32 device numbers. Do not use device numbers that don't exist. Repeat the device numbers in the table if you have fewer than 32 drives. The 4 data statements for the drive numbers are grouped into strings of 8 numbers each. Set the values in the strings accordingly. For example, the 6th digit in the 3rd data statement is the drive number for disk drive #22. Remember that only the E,R and $ commands in the editor section make use of this drive table.

A note about OverLink: The OverLink system genrally requires more disk space than one drive can afford. There is no restriction on running the system with one drive, other than the fact that you will have a very minimum system. It is recommended that you have at least two drives, with 3 being suitable for a large system using dozens of modules, in case of more OverLink library disks. As you will see from the OverLink SETUP files, the system allows for usage of large disk storage capabilities (ie. multiple SFDs, etc.).

2. Load the BBS. The BBS has two visible files in the directory, Just type load "?*",device,1 to load the first file. Make sure you copy the DarkTerm files to another disk before you use the terminal program. Follow the prompts as you would with the V3.0 system, outlined in that manual. The BBS will prompt you for disk #2 at the end of the load. Place your working copy of disk #2 in the boot drive. The BBS will attempt to load all the aforementioned files. If any are missing, you will find that you will not get any farther into the startup procedures. The BBS MUST load from a 1541 or compatible drive.

The order in which the setup works is slightly different from V3.0.

SYSTEM DRIVE. Normally with V3.0, the SYSTEM disk (or FILES disk) was set to drive 8:0, and could not be changed. This is the drive where the relative files, and system files reside. You now have the option to move this disk to any drive on the system. If you want to leave the system drive as 8:0, just hit RETURN. Otherwise, enter the device and drive number of the new system drive in the format "XX:Y", where XX is the device number (TWO digits) and Y is the drive number (ie. use 08:0 instead of 8:0). Make sure your files disk is on that drive when you are prompted for the files disk at the end of the start-up procedures.

LINK FILE. This option allows for external program chaining to the main BBS for expansive and radical changes to the operation of the BBS program. Details of this option can be found with any module that makes use of this feature, No explanation is given here. So, just enter RETURN.

MODEM FILE. See the BBS V3.0 Manual.

USE INVERTED CARRIER?. Some modems, such as a Hayes compatible (ie. SVC 1200) have a reversed carrier detect signal on the user port. This means that it would normally detect a carrier where other modems would detect a carrier loss. For all modem files written for this BBS, a note will be added to indicate whether you should use inverted carrier detect or not. If you use a Hayes clone, just enter a "y". A sure way to find out if you have inverted carrier is to use the normal carrier detect, and wait for an incoming call. If the BBS resets from carrier loss, you should use inverted carrier detect. Enter "n" to set carrier detect to normal operation.

CHARACTER SET. See the BBS V3.0 manual. Note that V3.1 relocates character sets independant of the load address, unlike V3.0, which required at $E000 (57344) load address.

TEXT FILE.	See the BBS V3.0 manual.
SYSOP PASSWORD.	See the BBS V3.0 manual.
SYSTEM PASSWORD.	See the BBS V3.0 manual.
DAY,MONTH,YEAR,TIME.	See the BBS V3.0 manual.

OVERLINK. Enter RETURN to NOT use the OverLink feature. In this case, the BBS runs like a V3.0 system. Enter "n" to use NUMERICAL format. Enter "w" to use WORD format and the command word interface (CWI). Both of these modes are explained later.

At this point, you will be prompted for the master disk (#1). Put this in the boot drive, and hit return. This will complete the start-up procedure. You will be prompted for the files disk, then the BBS will begin operation. From this point on, reference the V3.0 manual.

The OverLink Expansion System

In your BBS manual, the technical notes section mentioned that the BBS has the facility for expansion via the GameLink Module System. The term "GameLink" was dropped, and is henceforth denoted "OverLink".

The OverLink is a system by which multiple independant programs can run in addition to the normal commands found in the basic BBS system. The OverLink is so termed because of the 2 ways by which the BBS can interface to this facility. The system will work either by using overlays or program linking, or in some cases, will use both methods at the same time. Overlays are done before the BBS begins system initialization, and are used for replacing or modifying entire sections of the BBS BEFORE it is run. In this way, some parts of the BBS that you have little use for can be replaced with something the basic system does not have.

Generally, program linking is done AFTER the BBS has run. These are the individual OverLink program modules that are loaded into memory via the OverLink command on the BBS. These link files may be anywhere from 1 to 50 blocks long, and may be chained together to give very large programs. The typical link module is usually only 1 to 10 blocks in length. The link modules may or may not modify the main BBS while it is running. If it does do so, then the module will tend to have overlay code in itself that overlays parts of the main system. This is a very powerful feature. A BASIC BBS can not modify itself while it is running, unless files are overlayed of course. This involves too many restrictions on that type of system. On BBS V3, the code will be changed while it runs, and will usually change back to original format prior to the next incoming call.

The primary use of OverLink was originally intended for the use of making on-line games that can be played over the modem. Some of you may have heard of a BBS program called C-Net 10, which allows on-line games to be used with overlays. These games are restricted to a very small size (<16 blocks), and are written in BASIC which we know is slow. With our system, all games will be written entirely in machine language, and can be as large as 50 blocks without secondary chaining of the main BBS. Because of the nature of the OverLink, it is not possible for you, the user, to write your own modules. However, if you have an interesting idea, or even a program you wrote for C-Net that is PD, then don't hesitate to let us know. We are always open to suggestions.

The OverLink disk you purchased is not copy protected, and you should make up many backups, and keep the original disk itself untouched.

All OverLink modules are (C), whether it is based on your ideas or ours. OverLink disks will be sent to you for a minimum fee, as they become available. You will be notified of further OverLink disks, provided you are a legal owner of the BBS, and are on our Node list of DarkStar BBS3 systems. Appropriate documentation will be supplied with each OverLink Library disk.

The first library disk does not really contain a wide array of game modules. This is because of :

a) OverLink must first make an impact on the majority of users of this BBS program. The time and effort required to create game modules that go beyond the simple levels of BASIC game routines on other BBS programs depends on just how much they will be used, AND how much time the actual users of your BBS system will devote to these game programs. Once OverLink has made on impact and users are interested enough to the point of requesting games for us to create, we can be assured that it is worth our effort.

b) Time did not permit us to start writing any game modules. We were only able to get the first 14 modules completed by the official release date. Game modules are of less importance than the other system modules on the O-L disk, and as such were not considered mandatory for the first disk. Of course, the release of future game modules depends on just how well the sales of the initial V3.1 OverLink system go.

The 3 games modules on the first disk are written for the color system only. In particular, BlackJack was written with color in mind. Cursor plotting is a definite must with this module. Black Box is a simple game, a "timewaster", used as an introduction to just how effective cursor plotting can be. The games were not written in ASCII, since most users of the BBS run a color system, and we just are not aware of who is running an ASCII system. Please let us know if you would like these modules converted to ASCII. Because the modules are not copy-protected, appropriate BASIC binary file enocoder programs can be sent to you on listing for typing in special modules or for bug patches on current modules.

OverLink Expansion System

The OverLink routines exist on the BBS even if you run the system without the OverLink enabled. Obviously, there must be a way to access OverLink modules. The OverLink command will replace the VOTE command in the BBS, once installed. The VOTE command will not be disabled. VOTE will move from a main BBS command to an OverLink module, which happens to be module #1. See the docs on OLINK Module 1 for more information. As such, the VOTE command definition should be changed from VOTE to something more appropriate (ie. EXP for system expansion command). If you consult the BBS manual under VOTE, you will find that the OverLink (henceforth termed O-L) works in a similar manner. We will use the EXP command definition to represent the O-L interface.

When a user enters the command EXP from any of the main command prompts, the BBS will display a sequential description file for current O-L modules. This file is named "+gl" and must be placed on the files disk (drive 8:0). If a user enters EXP followed by a number from 1 to 99 (ie. EXP8), then the BBS will load and execute the O-L module designated by the number the user input. O-L modules have a file name format of "+ga.XX" for the ASCII BBS, and "+gc.XX" for the CBBS. For example, EXP8 would load and execute the module "+gc.08" when using the color BBS, and "+ga.08" when using the ASCII version. Load times depend on the length of the module.

Let's try an example O-L structure for the BBS. We have 5 current O-L modules, and we can use any number sequence we want. That is, we do not have to use 1 to 5; we can use 1, 4, 8, 56, and 99 if we want to, but we'll use 1 to 5 in our example. We'll be using the CBBS, so we'll use the color BBS "+gc.XX" modules. The following steps are done to setup our system:

1. Place the O-L modules on the OverLink disk. You may use the SETUP files to change the O-L module before it is placed on the O-L disk. Refer to the individual O-L module docs. The OverLink disk is the system drive that physical directory #8 uses. In the CREATE.CNF program, you had to set up the physical and relative directory device and drive number tables. The last one (the 8th) is now designated as the OverLink BBS disk. You may share other system files on this disk. If you are using the 8th directory, then you will have to share it with the O-L modules. Remember that this is physical directory #8, NOT relative directory #8. Set the device and drive numbers to the drive you want. For our example, we'll use drive 9:0 as the O-L disk. Use CREATE.CNF to perform this step, then use the SETUP files on you O-L module disks to change the modules, or simple copy them as is to the O-L drive. O-L modules, as they are when you first see them on the module disk, have a file name format of "olink.mXXX" or "olink.mXXX.a". The latter name is the ASCII module format, the former is for the CBBS. You can NOT interchange them. You should always make any changes to the modules on a backup disk, then copy them over to the O-L drive and change the file names to the appropriate O-L number. The XXX in the file name is the module volume number. For example, the VOTE command O-L module is module #1, and appears on the O-L module disk as "olink.m001" (color) and "olink.m001.a" (ascii). The setup file is called "setup.m001". On the O-L BBS drive, we will have to rename the file "olink.m001" to "+gc.01", once copied to the drive. Note that we do not use "+olink.m001.a", since we are not using the ascii version in our example.

2. Setup the "+gl" description file. Our file will appear as follows. Remember to write this to drive 8:0.

```
Expansion Command              Enter
------------------------------ -----
Voting Section................ EXP1
BBS Advertising Section....... EXP2
Las Vegas Blackjack........... EXP3
Hangman....................... EXP4
Trivia Query #1............... EXP5
This List..................... EXP
```

The above O-L modules would have file names of +gc.01 to +gc.05. If we ran the ascii version, the file names would be +ga.01 to +ga.05. Remember that the VOTE command is now the EXP command (or whatever you want for the command word, using CREATE.CON).

OverLink Expansion System (P3)

We now have the overlay files on the O-L drive, have changed the VOTE command word to EXP, and have created the "+gl" description file (remember that our overlay modules are on drive 9:0). We now boot up the CBBS using the normal procedures outlined in the manual.

Once the BBS is running, a user would enter "EXP" and would be given the description file on drive 8:0. If the user enters EXP5, then the BBS will load and execute "+gc.05", which should be the trivia module, from the O-L drive, which is 9:0 in our case. If a user enters an invalid number, such as EXP89 when there is no "+gc.89" file, the BBS will still attempt a load, but will not execute anything if there is no file. Therefore you do not have to have 99 modules running to use the O-L. Note that the BBS uses the Kernal LOAD routine, and only checks the routine exit status and not the disk error channel, so you will see the error light flash if a module is not found. Ignore this; it is harmless, and will disappear upon the next access to that disk drive.

Note that the OverLink module numbers do not have to correspond to the EXP numbers. For example, the VOTE module is module #1, but it can be accessed with EXP56 with a file name of "+gc.56" if you want it as such. Some modules can even be duplicated on the same disk, like the trivia module, which allows as many trivia sections as you want using muptiple copies of the same module via it's setup file.

That is about all there is to the actual OverLink system itself. Because it is always expanding, documentation will appear as more modules are created. Each module itself will also contain all necessary information for its use on the BBS. To finish up, you should take a look at the Command Word Interface, which follows this. It explains how to change the O-L routines so that you can use real command words instead of numbers (ie. TRIVIA instead of EXP5).

The Command Word Interface (CWI)

The CWI is an overlay program that allows the OverLink to run using command words, rather than the numerical selection mentioned previously. There is 1 file on O-L disk #1, named "cwi.com", that you will need to create a CWI command file.

This is not a run time overlay, and as such is installed along with the OverLink at bootup time. Refering back to the section on OverLink setup, you should install the O-L when the "OverLink:" prompt shows. As said before, you enter either "n" for numerical operation, or "w" for using the CWI system.

The "cwi.com" program is written in BASIC, and is very small and simple. It is used to create the command word data file. This program MUST be run on the OverLink Drive disk, the one you place the overlay programs on, which would be the physical directory #8 drive. If you intend to place a lot of other system files on the O-L drive, then you should rearrange the directory so that the command word data file is at the top of the disk directory. A program called Yellow Pages has been placed on O-L disk #1 for your use. It is a P.D. directory organizer, and does a great job of rearranging. For example, if your O-L drive is drive 8:0, then you should place the CWI data file just under the system counter file, "+sf". The CWI data file is named "++xc", and in this case, it would be the second file in the directory.

Load and list the CWI.COM program. Line 1040 sets the number of O-L commands, which would be the number of data statements used, one for each command. In the example given, VOTE and BBS are the only 2 listed. There are 2 commands, thus NC has a value of 2. As you add more commands, increase the value of NC as necessary. The syntax for command word definition is the same as that listed under CREATE.COM, except (1) the command word length can be from 1 to 8 characters, and (2) there is no length data byte in the CWI.COM program. All commands are divided by a chr$(128), and saved in sequence.

When writing overlays to the O-L drive disk, you do not save them in the format "+ga.XX" or "+gc.XX". You now save the file in the format "+-COMMAND". For example, VOTE would be stored as "+-VOTE", and BBS as "+-BBS". Remember! Only 1 to 8 characters. Going beyond this limit will cause problems.

From our previous OverLink example, we used EXP as the command redefinition for the O-L interface. With CWI, we get the same result when we type "EXP". We get a listing of the commands to be entered. Consider this a help file, like any other (the "+gl" file, that is). No need to enter EXP1 for the voting section; now we just enter VOTE, and we get the same result. The big difference is that when we enter an O-L command through the CWI, the BBS will access the "++xc" file from the O-L drive, and search for the command, and if found, load in the overlay and execute it, based on the name of the command. This is the reason why you place "++xc" at the top of a directory. Access times will be very quick when doing it this way. As you add more commands to "++xc", the access time will go up, but not enough to take notice. The most important thing to remember is that ALL O-L commands accessed via the CWI can only be accessed from the MAIN MENU prompt. You can not type "TRIVIA" at the main message command prompt, and expect to get the trivia module.

Well, that's about it for the CWI. Additional information may be given with each individual OverLink module. The CWI does not have a module catalog number like normal O-L modules.

Module #001 : VOTE Section Replacement
Files : SETUP.M001 - OLINK.M001 - OLINK.M001.A

With the OverLink interface replacing the VOTE command in the BBS, the first module to design would logically be one that replicates the VOTE command of the BBS. Thus, we have the VOTE section module. Funtionality is the exact same as that with the normal BBS VOTE command. Refer to the BBS Manual for details.

The SETUP file will allow you to change the drive to which you will place the "+vt.XX" files on. The "+vo" result file is still on drive 8:0, and the +VT.XX files can be on any drive this time around. The following lines are the only ones you may modify in the setup file:

100. DV : Device number of drive to place +VT.XX files on.
DR$: Drive number of drive to place +VT.XX files on.
V : Version of module to modify. Set to 0 to modify the color module, or 1 to modify the ascii version.
120. DF$: Destination file name. This usually defaults to the original file name, in which case a modification will result in the original file being scratched and saved over with the new file. You may change this to any file name you want, and if you use the CWI, you should change it to the command word, preceded with the "+-" characters. You should also change the destination drive too (see below).
130. Some characters in prompts can't be expressed in text strings, so it is necessary to use variable strings. Such is the case with q$ and r$, for quotes and the return character. Other modules may have more of these things found on this line.
140. Vote Selection Prompt. All modules will have at least one prompt to redefine. Unlike the CREATE.TXT program, you must change the string value here. Keep it less than 255 characters. You may have to concatenate if you use more than 80 characters. This modification should be fairly obvious to you; if not, don't modify the prompt.
250. This line sets the device and drive number for the destination drive, where the modified module is to be written to. You can change the drive and device number if you want; it will save placing things on the original OverLink library disk (always modify modules on a backup, never on the original disk purchased!).

You will then run this program. The old module is read in, the destination file is scratched, then the modified file is written to the destination drive. Be careful not to change the data statements. They are the machine language routines for reading and writing the module quickly from and to the disk.

Note that the original VOTE modules have defaults as you see them in the setup. You don't have to use the SETUP program if you do not want to change anything.

Changes to the VOTE command: you will remember that the command VOTE by itself reads the "+vt.00" description file, and VOTE12 would read and execute the vote procedures for vote topic number 12 ("+vt.12"). Well, in the O-L version, the prompt states the change. Entering "?" will give the "+vt.00" file, whereas entering a topic number directly does the same thing as the old command+topic number. For example, instead of having to use VOTE12 as in the old command, the O-L version would require only "12" as input.

All tabulations are done through the OverLink module as well. The results, and the summary that you saw with the "V" command in the editor section is rerouted into the module. The "V" command in the editor section is no longer active. Immediately after the vote topic is selected, and the ballot made, the level "s" user will be placed into the "V" command of the editor section automatically.

Module #002 : TRIVIA Section (P1)
Files : SETUP.M002 - OLINK.M002 - OLINK.M002.A

This is the first of 2 trivia modules provided on the library disk, This module does not have to function as a trivia module however. The basic design behind this program is that a user is given an introduction to a questionaire and allowed to answer a number of questions, the results of which are added to an answer file. For description purposes, we will express this module as a trivia section.

Like many of the modules you see on the library disk, this is the first of modules which can contain "cloned" sub-sections. This means that if you ran just 5 sub-sections, then each sub-section would operate as if you were only using one section with no-subsections. The only difference between one section and multiple sections is that you have to go one extra step to creating a sub-sectioned module.

The trivia section can be considered "cheat-proof" by the way answers are stored. A user's name and ID# are saved every time the module is accessed, provided the user agrees to answer the questions. Thus, you can see if a user attempts to answer questions more than once by looking in the answer file for the user's name to occur more than once. The basic operation of the module works like this:

1. If there is more than one trivia section, then a master menu appears describing each of the trivia sub-sections. This file is named "+t/menu", and is a standard SEQ text file you create. The user then enters the number of the section to access. Entering "?" re-displays the main menu. Entering RETURN exits to the main BBS.

2. The introduction file for the trivia section is displayed. If there is only one section, this file will appear in place of the above menu. This file is named "+ti/xx", where xx is the trivia section number, from 1 to 25. If you use one section only, the section number defaults to 1. For example, the intro file for section 5 would be "+ti/05". This file should outline the nature of the questions the user is to answer.

3. The user at this point has 2 choices. The prompt "Continue?" will be displayed. If the user replies yes, then his name and ID# are added to the trivia answer file for that section. The user will then proceed to answer the questions. If the user replies no, then he will be returned to the main BBS without any change being made to the answer file. If the user is a level "s" user, he will be asked to see the answer file for that section, then he will be placed back into the previous menu. Note: if you use 1 section only, exiting the trivia section exits the module, and you are returned to the main BBS. If you have more than 1 section, then you are returned to the main menu.

4. The question file will now be displayed. All questions are stored in one file, and are separated by a delimiter character, which is the "@" character. This character is used by other modules too, so become familiar with it.
As such, this character cannot appear in the questions themselves. Similarly, the "&" character cannot be used either. This character marks the end of the text file. Always end your file with the "&" character. Always place an "@" at the end of each question. A description of character delimiters will not be given. The following would be what a typical question file looks like:

```
What Is Your Name?@
What Is Your Quest?@
What Is Your Favorite Color?@
This Is The End. Bye!&
```

As you can see, the "@" is used after each question, and the "&" is used at the end of the file. Remember, only 70 questions maximum per section.

Because the screen clears after each answer and question, you can use a lot of color in your question files, provided you use the color version of the BBS.

After each question is displayed, a user is allowed to enter from 1 to 39 characters for an answer. If the user hits RETURN alone, the question file is aborted, and all answers made up to that point are saved to the answer file. If no questions were answered, the section is aborted altogether. But the user's name is still saved to the answer file to let you know that he attempted to answer the questions. Note that the question file is named "+tq/xx", where xx is the trivia section number. The format is like the intro file, except a "q" is in the file name rather than an "i".

Module #002 : TRIVIA Section (P2)

5. Once all questions are answered, the answers are saved to the answer file, which is named "+tr/xx", where xx is the trivia section number. Note that for each section, you must have an intro file, a question file, and a result (answer) file. The result file should be created with any starting text you want. When answers are stored, they are appended (added) to this file. Thus the file grows and grows, and if you are not careful, you could end up with a very large answer file. The module prevents any access to the trivia section if less than 10 blocks are free on the disk. The user will be notified of the low disk space and will be returned to the main menu (or the BBS). The answer only is saved to the file. If a user answered 5 questions, the following would be written to the answer file:

(User ID#) User Name

Answer 1
Answer 2
Answer 3
Answer 4
Answer 5

Make sure your questions are simple. Only 39 characters + return are allowed per answer, so multiple answer questions would not be a good idea. The module does not verify correct answers. This is your task. Use the other trivia module if you want the answers to be checked if they are right or wrong.

6. After the answers are written, the user will be returned to the main menu (or the BBS), unless that user is a level "s" user. If so, he will be asked to view the result file. The option to clear the answer file is also available. You will have to clear the file frequently if you have limited disk space.

Using The SETUP File:

100. V : Version of the module to modify. 0=color 1=ascii.
120. DF$: Destination file name.
300. Set the device/drive for writing to another disk.

Type LIST5000- to see the DATA statements that have to be set for this module. Most modules have their parameter data from lines 5000 onwards.

Lines 140-190 contain the text prompts for the 6 prompts this module uses.

5000. Connect Time Flag. Set to 0 if you want a user's connect time to stop during his stay in the trivia module. Set to 1 otherwise.
5005. Number of trivia sections. Select a value from 1 to 25.
5010/5020. Device and drive number for the main menu. If you use more than 1 section, then this is for which drive the "+t/menu" file appears on.
5030-5080. Device/drive numbers for each section. Each trivia section can have its question, result, and intro file on its own drive. This allows for systems with large disk capacity. The device and drive numbers are set in the same manner like the table the INSTALL DRIVES program uses, except that it is only for 25 drives, not 32. you CAN and should try to keep all sections on the same drive for readability.

Module #003 : Advertising Etc.
Files : SETUP.M003 - QLINK.M003 - QLINK.M003.A

This section has no pre-defined use. Originally, the first idea for this module was to use it for advertising BBS numbers. The concept of this section is fairly simply. Each sub-section of this module has a file, which grows as data is addded to it. For example, when users compile a BBS numbers list, they will add the BBS name, number, etc. to this file. All data is appended for chronological ordering.

A user basically has 2 options when accessing a sub-section. The user may use (R) to read the data file, or use (A) to add to the data file. In either case, one or the other option will be accessed in turn until the user enters RETURN. The data file is a simple SEQ text file, that must initially contain at least one character. Generally, you should add a simple description of what the file contains, and what the user should add to the file if he should choose to do so.

This module sub-divides into 25 sub-sections in the same manner as the previous trivia module does. You may have up to 25 sub-sections, or just one sub-section. As with the trivia section, you will have to create a master menu if you want to have more than one section. This file is named "+a/menu", and the context of this file should follow along the lines of "+t/menu" in the trivia section. See module #2 docs for information on how the sub-section system works.

For each sub-section (or for just one section if chosen), there is only one file. This file is named "+++bbs.xx", where "xx" is the section number. For example, bulletin ad section 1 would be named "+++bbs.01". Always create this file intially with at least one character contained within it, to prevent the module from adding to a null file. From the main menu list, the user may enter "?" to re-display the section numbers and their descriptions given in the "+a/menu" file. If you have one section, you will proceed straight to section number 1. The user will then be given 2 options: (R) to read the ad file, or (A) to add to the file. Entering RETURN places you back to the main menu, or to the main BBS if only one section is used.

The READ option is fairly simple, so it won't be explained. The ADD option involves adding new text to the file. Depending on your needs, the sections could be used for adding BBS numbers to a numbers list, or having users add wanted/for sale ads to others. The choices are yours. When adding text to the file, you will be allowed to add only so much text at a time. Twenty blocks must be available on the disk drive for the current section to be able to add to the +++bbs.xx file. If you are using the color BBS, the module will allow you to enter text with either a color editor or a line editor. The ascii version only allows line entry. In either case, the editors are one-shot only, meaning you can only enter text as you go, and not re-edit upon exit.

When using the line editor, a RETURN alone on a line exits editing. The color editor uses STOP to exit. When text has been entered, the user may abort at this point. If not, the text will be added to the data file for that section. Then user will then be placed back to the point where he may re-read or re-add to the file. The remainder of this module is explained under the setup procedures.

The SETUP file. Most of the parameter data exists from lines 5000 onwards, with a few exceptions.

100. Set V=0 for the color module, or V=1 for the ascii module.
120. DF$: Destination file name.
140-190. These are the text prompts for this module.
300. You may change the destination device and drives numbers here.

5000. Connect Time Flag. Set to 0 for users' connect time to stop during the module access. Set to 1 for normal time.
5010. Number Of Ad Sections. A number from 1 to 25.
5020/5030. Device and drive number that the "+a/menu" file resides on. Ignore this for one section usage.
5040-5060. Access Levels. The ad module runs separate from the BBS in terms of access levels. You may set the level for each of the 25 sections. Any level may be used, from 1-9, or "s" for a SYSOPs only section. There are 25 numbers in the data, one for each section, ordered from left to right, top-down.

Module #003 : Advertising Etc. (P2)

5070-5090. Text Entry/Display Mode. Each section can have from 1 of 4 settings to it's mode of operation. The following values allowed are:

0: Line Entry, Non-Continuous. All text is entered as line input only. No color graphics are allowed. Non-continuous refers to the actual reading of the text with the (R) option. The "@" delimiter character is added to non-continuous files to allow the display to break at the point an "@" shows up. In line entry mode, the user will be prompted to hit RETURN to go to the next ad. With color mode section, the cursor will flash between ads and wait for the user to hit RETURN. The "@" character may NOT be used in text entry by a normal user. The module will prevent this from occuring. The level "s" user may use "@" however, when entering text. This allows for forced breakpoints in the data file. Note that the only way to break out of the ad file display is to use the Xon-Xoff character for list display abort. For example, the user enters "a" to abort during the file display, not between ads. Every file displayed in all modules of OverLink will be abortable using the standard list control keys, unless otherwise stated.

1: Color Entry, Non-Continuous. Color entry allows you to enter text in the same manner as the BBS color editor does. You do not have any editing capabilties however. When the number of characters entered exceeds the set limit (see limits below), the cursor will flash to let you know you have entered the limit, and you will have to hit RETURN at this point to save the text. Non-continuous mode works as described above.

128: Line Entry, Continuous. Same as 0, but with continuous mode enabled. This allows the file to be read without any stops. The "@" delimiter is not used in this case, and the user is free to use it in the text itself.

129: Color Entry, Continuous. Same as 1, but with continuous mode.

5100-5120. Line/Character Limits. For each section, you must set a pre-defined limit to the amount of text that a user may input. The maximum limit for color entry is 2048 characters. For line input, it is 50 lines. The number for each limit in the data statements will be either a character count, or a line count for the limit. Be sure that you align your mode values with these so that color modes align with character limits and line modes align with line limits. For example, the default sets all 25 sections for line mode with a maximum line input limit of 5. The SETUP file will split the number into hi/lo byte values, so that entering a value of 1560 for a character limit in color mode is valid.

5130-5150. Add Name Mode. You may choose to add the users name and ID# to each ad file, be it color or line. This will be added to the +bbs.xx file before the text is written to it. A value of 0 will add the user's name and ID# to the file, a value of 1 will suppress it.

5160-5210. Device/Drive Numbers. As with the trivia module, each section has it's own drive and device number. See the trivia module (#002) for details on drive table setups.

Module #004 : Bulletin Section Expander
Files : SETUP.M004 - OLINK.M004 - OLINK.M004.A

This module is provided as a faster more simple way to archive your bulletin section. If you have a lot of bulletins that you want to make as permanent fixtures on your BBS, then use this module and place them in one of it's sub-sections. It is possible to store an un-limited number of bulletins with this module. The program acts as a simple file reader, with no limit on how many bulletins can be accessed. There is no relative file for this module. As with modules 002/003, this one can be split up into 25 sub-sections. each with it's own disk drive.

When the user enters this module, the master menu will be displayed IF there is more than 1 section enabled. For 1 section use, this part will be skipped, and the user will proceed to section 1. The master menu is named "+b/menu", and follows the same guidelines as the previous 2 modules do. Entering "?" will re-display this file. Entering the section number will allow the user to proceed to the appropriate section. Entering RETURN will exit back to the main BBS.

Each section has it's own main menu, which is named "+b.xx.menu", where "xx" is the section number. For example, section 20 has a main menu name of "+b.20.menu". The main menu is a SEQ text file as are all bulletins. Because there is no structure to this BBS module, like there is with the bulletin section in the main BBS, you are free to divide and sub-divide your bulletin section at will, using text files for sub-sub-menus etc. The manual mentions how primate text mode in the bulletin section works. This module operates the same way, except that you may have 25 "discrete" sections as well. Once in a section, "?" will display the main menu. RETURN will exit to the master menu, or to the main BBS in case you only use 1 section.

Bulletin names can be up to 10 characters long, and can use any characters you like, except for upper case. The bulletin names are set up so that one section's bulletins will not interfere with another's. That is, you can have a bulletin with the same name in different sections. Bulletins exist in the format "+b.xx.NAME", where "xx" is the section number, and "NAME" is the actual name of the bulletin. For example, a bulletin called "intro" in section 4 would be named on disk as "+b.04.intro". The important thing to note here is that unlike the BBS bulletin section, there is NO padding of the bulletin name with spaces to make it length 16.

The SETUP File.

100. Set V=0 to modify the color module, or to 1 to modify the ascii module.
120. DF$: Destination File Name.
140-150. Text Prompts For This Module.
260. Line to change destination drive.
5000. Number Of Bulletin Sections. A number from 1 to 25.
5010-5030. Access Levels. Set to 1-9 or "s" for each section.
5040-5060. Connect Time Flags. Set 0 for connect time to stop, 1 for it to run normal, during module access. Note that you must set connect timers for each section.
5070-5090. Display Main Menus. For each section, you may have the main menu file displayed as soon as the user enters the section. This is analogous to the user entering the section and typing "?". Use a value of 0 to display the opening menu, or 1 to disable it.
5100-5110. Device And Drive Number For The Master Menu. This is for the file "+b/menu". For one section only, this file is not used, nor is this setting.
5120-5170. Device And Drive Numbers For Each Section. See module #002 for details.

NOTE: A late addition was made to this module. If a user appends the asterisk (*) character to the bulletin name, the BBS will initiate a Punter file transfer of that bulletin. There is no provision for Xmodem bulletin transfers. For example, if a user normally enters BBSNUMS as a bulletin to read, BBSNUMS* would cause the file to be downloaded as a single file Punter transfer, rather than a normal bulletin read. This feature can be handy for a user who wants to archive your bulletin section.

DarkStar BBS V3.0 - The OverLink - (C) 1986 D.S.S.

Module #005 : On-Line File Copier (LOCAL MOD)
      Files : OLINK.M005/A

This is an exact duplicate of the 2 drive file copier on disk #2 of the BBS, named COPY.FILE, with one exception. The source and destination device numbers can range from 6 to 19, not just 8 or 9.

See the BBS manual under COPY.FILE for more information on how to use this program. Note that this module will not be executed if the user on-line is not the SYSOP (ID#0). Although the program module will load if the command is entered, the program will abort back to the main BBS if any other ID# is found. This module runs in local mode, and cannot be executed from remote, this being why the SYSOP-only feature was added.

Since this program must store the files to be copied in memory, it was necessary to find an area that wasn't in use. Unfortunately, the BBS uses virtually all of memory, so the only way to make room was to temporarily use part of the ISFS data tables. The user name table is used as the files buffer, which means that the users are erased from memory during the operation of the copier. When you enter "X" to exit the copier program, the "Users." section of the ISFS setup routine will be executed to restore the users to memory. Once complete, you will be returned back to the main BBS.

As you can see, there is only one file for this module. The /A ending on a module file name indicates that it is to be used with either BBS. It contains auto-detect routines to determine which BBS is running. There is also no setup file for this module. It is a straight copy from the COPY.FILE program with the aforementioned modifications.

This file copier allows you to copy from drives that have a device value as low as 6 or as high as 19. You CAN have a disk drive with device 4-7, if you do not have a printer hooked up.

DarkStar BBS V3.0 - The OverLink - (C) 1986 D.S.S.

Module #006 : Relative File Maintenance Manager
Files : OLINK.M006 - OLINK.M006.A

You may have read in the manual under relative file editing that some fields in the records of the files could not be changed by any means other than a disk doctor. This program allows you to change every byte of every record of all 4 system relative files (NOTE: Please read the section in the BBS manual under record structure to understand how to change each field in the records of the files). This program is restricted to level "s" users only. With any of the level "s" OverLinks, it's best to make the numbers of the OverLink command (or the command value using the CWI) unreferenced to keep the normal users from knowing these sysop commands exist.

The end of this documentation summarizes all the field descriptions and definitions for all 4 files. The program consists of 2 menus. The first menu consists of the relative file selection. Due to relative file "bugs", the files will be opened and closed for each record operation. Selecting the file will place you in the second menu. The second menu consists of 4 basic operations: ADD/DELETE/MODIFY/SUMMARY. Each command loops around, waiting for records to work with until you enter return to exit to the main menu.

When adding a record, you must enter all data for each field, or the command is aborted. When using the modify command, the current field values are echoed, and new input is required after that, UNLESS you enter a return by itself. By entering a return at any prompt using the modify command, that field will remain unchanged, and you will be able to continue onto the next field. For delete, modify, and add, you will be always given a second chance to confirm your options, in case you make a mistake.

Since all operations are done on the disk level only, it is necessary to reset the data to memory. So when you exit the program, the ISFS tables will reset. Some records are stored in upper case format. This does not mean you have to hold the shift key down. All input is properly converted regardless of your input. Sometimes the input allows you to go beyond the norm. That is, it is possible to add erroneous data as input. Memory was at a premium, and it was not possible to verify all input to be correct. Just like with using a disk doctor, you can add the wrong input. If this happens, just abort or re-edit. The following is a list of all fields for the records of the 4 files, with descriptions of the abbreviated 3 character prompts, and the input expected.

Relative File #1: Messages

LVL: Message access level. Input: 1-9.
RF#: Message reference number. Input: 1-65535.
FID: User ID# of message writer. Input: 0-255.
TID: User ID# of who the message is addressed to. Input: 0-255.
PVT: Message public/private status. Input: 0 for public, 1 for private.
ALL: ALL status. Input: 0 if message addressed to all, 1 if addressed to a user.
CAT: Message category number. Input: 0-9 (or the number of categories you have, minus 1).
SUB: Message subject. Input: 1-25 characters.
DAT: Date message was posted. Input: DDMMYYHHMMx. DD: Day MM: Month YY: Year HH: Hour MM: Minute x: "a" - am or "p" for pm. You must make sure the length is 11 characters. For example, Jan. 5, 1987, 12:34 am is "0501871234a".
FWD: Forward flag. Input: 0 if message not forwarded, 1 if it is.
FWI: Forward ID#. Input: 0-255, the ID# of the user who forwarded the message.
REP: Replied counter. Input: 0-255, the number of replies made to the message.
RRF: Replied reference number. Input: 1-65535, the reference # of the message that was replied to, OR a 0 to indicate that the message was not a reply to any other message.

Module #006 : Relative File Maintenance Manager (P2)

Relative File #2: Users

LVL: User access level. Input: 1-9, or "s" for sysop level.
NAM: User name. Input: 1-25 alphabetical characters, plus spaces or the "." character.
PWD: User password. Input: 1-8 characters.
DAT: Date of last logon. Input: See DAT under message relative file description.
TLI: User time limit. Input: 1 to 59 minutes.
TUS: Connect time used. Input: 0 to the user time limit byte above (This byte applies to daily time limit only).
TST: Time status. Input: "YN" binary input, consult manual under the EDIT.USERS program for details.
RDA: Relative directory access. Input: the 8 "Y"/"N" binary input string. See EDIT.USERS and CREATE.CNF.
PDA: Physical directory access. Input: Same as RDA.
SAC: System accesses. Input: Same as RDA and PDA.
RF#: Hi message reference number of last message read. Input: 0-65535.
TUP: Total Uploads. Input: 0-65635.
TDN: Total downloads. Input: 0-65535.
MTO: Messages written to user. Input: 0-65535.
MFR: Messages from user. Input: 0-65535.
TLG: Total logons. Input: 0-65535. NOTE: On page 58 of the manual, this byte was erroneously set to the definition of the MTO byte. The TLG definition is the correct one.

Relative File #3: Programs

DIR: Directory number. Input: 0-7.
FID: ID# of user who uploaded the program. Input: 0-255.
NAM: File name. Input: 1-16 characters.
PWD: File password. Input: 1 to 8 characters for a password, OR the left arrow symbol for no password ("aaaaaaaa" will indicate no password. The left arrow key places 8 "a"s in the field).
DAC: Download accesses. Input: 0-999.
SIZ: File size. Input: 0-999.
TYP: File type. Input: "p", "s", or "u".

Relative File #4: Bulletins

LVL: Access level of bulletin. Input: 1-9.
L#1: Link number 1. Input: 0-254. See EDIT.BULLETINS.
L#2: Link number 2. Input: 0-255. See EDIT.BULLETINS.
NAM: Bulletin name, Input: 1-8 alphanumeric characters, plus spaces.
SUB: Bulletin description. Input: 1-25 characters.

DarkStar BBS V3.0 - The OverLink - (C) 1986 D.S.S.

Module #007 : Terminal Link V3
Files : OLINK.M007 - OLINK.M007.A

This program provides a terminal linkup so that you can log onto other BBS systems while remaining in the BBS operating environment. This module's functions are a subset of those in DarkTerm 4.0. The autodialing feature could not be provided, as there is no place to load a modem file into. You will have to manually dial out, or send out AT dialing commands manually for smartmodems. The function list here has the same definitions as those of DarkTerm 4.0. Therefore, only the command summary is given here. If you want further details on the command functions, you should consult the DARKTERM DOCS file. If you know how to use DarkTerm 4.0, you will have no problem with this terminal link. If you don't, then read those docs, as they will not be re-printed here.

Status Line:

```
12:34a  R:01 C:01 6144 ASCII COM B:H:I:U
  1        2       3    4    5  6 7 8 9
```

1: System clock.
2: Cursor row/column.
3: Buffer count. There is a 6K buffer in the term, not exactly the greatest, but when you still have a BBS in memory, you do what you can.
4: Terminal status. Will read ASCII or COLOR (as opposed to A and C in DT4).
5: Command mode indicator. Reads COM for command mode, or blank otherwise.
6: Buffer open/closed status.
7: Hide output status.
8: Modem I/O inhibit status.
9: Upper case lock status.

Functions:

Shift Run/Stop	Toggle color/ascii mode.
Shift Return	Toggle command mode.
SPACE	Pause/resume output.
STOP	Abort output.
C= B	Buffer Options (<P> and <E> not available).
C= C	Change terminal options: P: Toggle protocol B: Change block size D: change disk device number #: change disk drive #.
C= D	Disk commands.
C= F	File options (only <R> and <T> supported).
C= H	Hide screen output.
C= I	Toggle modem I/O inhibit.
C= L	Load character set ("rom" for rom set).
C= M	Modem options (2400 baud not supported).
C= O	Toggle buffer open/closed.
C= S	Set terminal colors.
C= T	File transfer options.
C= U	Toggle upper case lock.
C= X	Clear buffer.
C= (Pound)	Send ascii delete.

DarkStar BBS V3.0 - The OverLink - (C) 1986 D.S.S.

Module #007 : Terminal Link V3 (P2)

NOTE: Run/Stop Restore is still tied into the BBS, so a computer cold start will occur if you use it. This means that you should be very careful about not getting disk errors, or you'll have to re-boot everything.

This module can be used by the sysop only (ID#0), and since it is a local mode module, you should not make public notice that this module is present.

The C= X exit function will do 2 things. First, the program looks for the text file on the OverLink drive. This is the same text file you input when setting up the system initially. The terminal program overwrites most of the text prompt storage area, so it must be put back into memory. The text file must be named "+bbs/txt" and be put on the OverLink drive. After this is loaded back into memory, the IFSF tables will reset. The terminal uses the data table memory for the 6K buffer and for the screen swap storage area, so this was necessary as well. After this, you will be returned to the main command prompt.

OverLink modules like this one stress the importance of having a multi-drive system. The overhead here is 32 or 34 blocks for the module, and that used by the text file, which will be at least 12 or so blocks. The OverLink was not designed for a uni-drive system, and you will realize this when more modules are created.

Module #008 : File Section Xpander (P1)
Files : SETUP.M008 - OLINK.M008 - OLINK.M008.A - +-+M008 - +-+M008.A - LINK.M008.C - LINK.M008.A

This is one of the largest modules on the library disk. It serves to supplement the standard file section by adding several new commands, as well as to re-organize the current commands and directory structure of the existing file section. This module will link into the main BBS and take control of several of the main BBS input/output routines. Because of it's size, it is located in a different area of memory, which happens to be the user data table. During entry to this module, the load time will typically run about 16 seconds on a 1541. When exiting the file section, the BBS needs to reset the user name table, which is one of the 5 steps in setting up the ISFS data. This setup time depends on the amount of users you have. Typical times range from about 25 seconds (no users) to 45 seconds (255 users) on a 1541. If you are using an IEEE drive as the system drive, you can expect a significant decrease in reset and load times.

The new features of this section include:

-Xmodem Sum And CRC Protocols, Both ASCII And Binary With Auto-Pad Stripping.
-Optional Upload/Download Logs.
-Restricted Time Periods For Uploading/Downloading On Any Directory.
-Additional 8 Physical Directories.
-Multi-Downloading From Relative Directories.
-Forced Message Writing/Download Option.
-Information Files.
-The Ability To Read A File From A Directory.
-New Combined Directory Access Structure.

All these features will be explained in detail according to the new commands that use them. Note that the new module command definitions replace those used by the old file section.

Combined Directory Structure.

With the old file section, the physical and relative directories each have their own commands, and the listing, uploading/downloading commands all worked with 2 distinct directory formats. With this new section, both formats are grouped into one combined directory list. The relative directories will occupy the lowest directory numbers, and the physical directories will occupy the highest directory numbers. Certain commands for physical and relative storage will be shared with this new format. For example, say you have 6 relative directories and 5 physical directories. The new format organizes the directories as follows:

Directory	0 :	Relative	directory	0	Directory	6 :	Physical	directory	0
"	1 :	"	"	1	"	7 :	"	"	1
"	2 :	"	"	2	"	8 :	"	"	2
"	3 :	"	"	3	"	9 :	"	"	3
"	4 :	"	"	4	"	10 :	"	"	4
"	5 :	"	"	5					

Module #008 : File Section Xpander (P2)

The module will set the directory limit as the sum of the number of physical and relative directories, in this case being 11. The base directory is still 0, not 1. The module will be able to recognize the directory as physical or relative and assign the file commands appropriately. If you have no physical or no relative directories, the module will still operate with just one format as well as it would with two.

Xpander Quirks.

If a user drops carrier in the file xpander, the system will not disable auto-answer if you use a smart modem. During the reset period as the user table is read back in, a caller could log into the system. This is not a flaw. The carrier loss is detected before the system can get back to the main menu. Thus, anyone on the system during the user reset will be disconnected immediately without ever gaining real access to the system.

Xmodem downloads will time out to the file section main command prompt after 6 seconds if the user has not hit a key.

There is no multiple file upload command for relative directories. It is not possible to add multiple records to a relative directory at one time. Similarly, it is not possible to update download accessed for 50 files at one time. So an MRD command will not update the download access counter in the files downloaded.

It is a good idea to not disable the old file section. The idea behind this is based on a multiple sysop system. Say, for example, you had files that assistant sysops wanted to upload or download files that they do not want to let the users know about. If you have an upload or download log enabled, and have them set for user access, then those files will update the logs after the file transfer. The only way to get around this is to go to the old file section, perform the transfer, then exit. The logs do not update. Think of the old file section as a "quick" access file section for priveledged users. You should change the old file section command definition to a value that normal users will not be aware of. The Xpander resets the old file section to it's own mode of operation once a user exits from it.

The help file for the file section is still used by the new one. You are responsible for adding the new commands into the help file yourself, and also removing/modifying the old ones in the "+he" text file.

When using the D command to display the available directories, up to 16 descriptions are displayed. The old file section shared 8 descriptions with both directory formats. Physical directory 0 shared the same description with relative directory 0. The text prompts for the new file section include the 8 directory descriptions for the 8 physical directories. The 8 in the standard text file are now solely for the relative directories. You may have to change those prompts in the text file if necessary.

Also, the descriptions do NOT appear anywhere except when using the D command. This gives a cleaner appearance to the file section. You will not see the descriptions when typing U or D etc..

Module #008 : File Section Xpander (P3)

File Expansion Commands:

(D) Display Available Directories: Entering D alone will display the directories available and their descriptions. If a user does not have access to a directory, then it will not be displayed. This is a nice feature not found in the old file section. The user will be asked for a new directory number. When selected, the new directory becomes the current directory. Entering D followed by a directory number will set the current directory immediately without displaying the directory descriptions. If you have the "show-directory" flag enabled, the current directory number will appear next to the main file section command prompt. Attempting to switch to a restricted directory will not be allowed. Note that most commands can switch the current directory automatically without using the D command at all.

(PRO) Set Xmodem Protocol : The default Xmodem protocol must be set by you with the SETUP file. You may set it to one of the 4 values set with this command, and that setting will be effective as soon as the user enters this module. The protocols available are:

-Sum Mode, Binary
-Sum Mode, Ascii
-CRC Mode, Binary
-CRC Mode, Ascii

Details of Xmodem transfer can be found on most bulletin board systems. A detailed discussion of what Xmodem is will not be given.

This module uses automatic Xmodem file transfer. That is, the last block, which usually contains padding characters to make the block 128 bytes, is handled in a special way. This padding (usually CTRL-Z else 0) is written to disk along with the file. This can create problems with text files and binary files. The module will double-buffer the data so that the padding is removed before the block is written to disk. During an Xmodem transfer, you will see that downloads use 2 128 byte on-screen buffers. The upload command does not need to double-buffer, so only one screen buffer will appear.

As with Punter transfers, you may abort the transfer with the STOP key at any time. Unlike Punter transfers, corrupt files are removed from the disk if they should occur. This happens when a user disconnects during an upload. The file would normally be left on disk during a Punter transfer as a partial program. Xmodem will delete this file in the event of carrier loss, so that only complete files are uploaded. Note that Xmodem also has a timeout feature that will cause transmissions to be aborted in the event of bad data or line noise that can't be corrected. CTRL-X repeatedly entered after a manual abort will usually stop an Xmodem transfer.

(BBS) Return To Main BBS : It is very likely that a user may hit RETURN one too many times. This would have the effect of reseting the user data and dumping the user back to the main BBS. To remedy this, a separate exit command is used. If a user were to hit RETURN now at the main file command prompt, a prompt would be displayed to indicate how to get back to the main BBS.

(T) Toggle Directory Banks : This allows you to use expanded physical directories. Using the second bank, you can use up to 8 more disk drives for directory storage. But there is a catch. The new bank will act just like the old bank, except that it uses a new physical directory drive table created with the SETUP file. Think of the second bank as an extension to the current directory. For example, physical directory 0 would have 288 files of storage if 2 1541s are used, one for bank 1, and one for bank 2. Be careful of the bank distribution. If you have 10 physical directories you can place 8 of them in bank 1, and 2 in bank 2. But the physical directory size set with CREATE.CNF will be set to 8, so that the system will think that bank 2 will have 8 directories, when in fact you only would have 2. You then must set directories 2-7 in bank 2 to those of directories 2-7 in bank 1, so that they will not point to null directories.

Module #008 : File Section Xpander (P4)

A more logical way to handle the situation of 10 physical directories would be to group 5 in directory bank 1, and 5 in bank 2, then set the physical directory size to 5 in CREATE.CNF. This will not give any null directories. If you used 10 1541 drives for the directories, then each physical directory would have 288 files and 1328 blocks, split up between the 2 banks. The T command is really intended for a large system. If you don't want to use it, then set the length byte to 0 for the T command in the SETUP file. Note that T does not toggle the relative directories. Those will always remains as is. The module will also reset the bank to one upon module exit, in case it was left as bank 2. This ensures that the main BBS is using the correct physical directory drive table that was created with CREATE.CNF.

(DIR) List Disk Directory : The C (Physical) and LIST (Relative) commands are now combined to this command. The module will be able to list either a physical or relative directory with this one command. The function of this command is that of both C and LIST. See the BBS V3.0 manual for details. Note that adding a directory number after the command will switch the current directory and then list the directory for the new current directory. For example, using DIR6 is the same as using D6, then DIR. This command replaces both LIST and C.

(DOW) Punter Download : This command replaces both the DOW and D command of the old section. Once again, the 2 commands are combined. After a download, the download log may or may not be updated (see DL).

(UPL) Punter Upload : This command replaces both the UPL and U command of the old section. After the file transfer is complete, the upload log may be updated (See UL). Both DOW and UPL can change to a new directory by adding the new directory number after the command.

(MD) Multiple File Download (Punter Only) : This command replaces MD in the old section. It also adds the multiple download option to the relative directories as well. When downloading from a relative directory, the stored file information is derived solely from the records in the +PR relative file only. You MUST be sure that all files match to those on the disk itself, or the BBS WILL hang somewhere. The download access counters in the records will NOT update after a multi-rel download. Be aware of this fact. You will have to use the download log as an indicator instead of the counters if you need to know how many times a file has been downloaded. The download log itself will update after either type of multi-download, in a specific way that distinguishes it from single downloads (see DL). The mode of operation for the multi-rel download is the same as that of the physical method. They only differ in where they each get their directory information from.

(MU) Multiple File Upload (Punter Only) : This command replaces the MU command in the old section. There is one major difference in the new version. As each file name is received during a transfer, it is checked to see if it already exists on disk. If it does, then the file transfer is aborted. Thus, you now have an MU that can NOT crash the BBS. The module will check for bad characters, insufficient disk space, and file existence. You may now allow anyone to access this command without worrying about losing precious files. Note that there still is no multi-rel upload command for reasons explained earlier. Add a directory number to MU or MD to switch the current directory. The Xmodem commands below can also switch the current directory. Note: When setting up user access and directory setup for the file section, interpret the setup with respect to the old file section commands. It will make understanding the new file section easier. For example, the DOW command has 2 separate user access bits in their record, one for physical, and one for relative. DOW in the new section will interpret this based on both the DOW and D commands of the old file section.

Module #008 : File Section Xpander (P5)

(DL) Download Log :
(UL) Upload Log   :

Both the download and upload logs have the same method of operation, except that one is used in conjunction with uploading, the other for downloading. They will be explained as one unit. Both logs are optional. You may run one or the other, or both. Similarly, you can have either (or both) logs be user accessible or sysop level ("s") accessible only. The SETUP file will allow for your own configuration.

Upload and download logs will update when using the UPL, DOW, XDOW, XUPL, MD, and MU commands, provided the logs are enabled. When a single file upload or download has completed, the user will be notified that the log is being updated. You will have to create the log files initially, so that at least one character is in it for the upload and download data to append to it. Both logs have their own device and drive numbers for the file locations, which you specify in the SETUP file. The upload log is named "+u.log" and the download log is named "+d.log". Once again, these are simple SEQ text data files.

For single up/down-load updates, the user's ID#, the name of the file, and the date transfered are stored in one line in the format: FILE NAME / USER ID# / DATE. The date does not include the time of day, only the date is stored. Their is only one log for uploads and one for downloads. All directories, physical and relative, share the same upload and download logs. If you have files that you don't want to appear on the logs, then you will have to hide the files under new file names, or allow the elite group of users to access the original file section. Or, you can disable the logs from general user access by restricting the UL/DL commands to level "s" users only. In this case, when a normal user enters UL or DL, the BBS will ignore the command.

For multiple file uploads and downloads, all files transferred will be written to the log as one unit. The way to tell if a multiple file transfer has occured is to examine the file for the lack of an ID# and date next to a group of files. A multiple file update to either log writes the user ID# and date only to the FIRST file, and not the remaining files that were transfered. Thus, you would get a format that looks like this:

```
FILE #1 NAME / USER ID# / DATE
FILE #2 NAME
FILE #3 NAME
FILE #4 NAME
   Etc....
```

When the next file transfer occurs, the user ID# and date will show up again. Upload and download logs can be quite large on a BBS that is file transfer intensive. Using the user ID# and not the user name, and using the user ID# and date only once in multiple file transfers saves quite a bit of space and update time. The UL and DL commands do not have the option to clear the logs as does the LOG command in the user section. You will have to use the edit file and DOS wedge in the editor section to manually clear the logs while on-line.

If you disable either or both logs, then you do not have to have the files on disk. The BBS will not update anything after a file transfer, and will perform in the same manner as the old file section. The commands of the new file section are provided for the SYSOP who wants to take advantage of a file section that has large amounts of disk space to offer.

Module #008 : File Section Xpander (P6)

(XDOW) Xmodem Download : This command allows for single file Xmodem transfers. It is important that both the terminal and BBS use the same transfer protocols when up/down-loading (ie. CRC to CRC, and SUM to SUM, not CRC to SUM or SUM to CRC). If the protocols are incompatible, or if the user attempts to use Punter transfer, the BBS will time out after a dozen seconds or so, and abort the command. As mentioned before, a manual abort of the Xmodem transfer can be accomplished by hitting STOP and then waiting for the terminal/BBS user to enter CTRL-X enough times for the other system to recognize the cancel request. There are no multiple file transfers with Xmodem protocol. It should also be noted how the ASCII transfer of Xmodem works. For downloads the file stored on disk must be a PET ASCII file. If the file is stored as ASCII, you should select a binary tranfer, so that the data will not be translated. If the file is PET ASCII, it will be converted to ASCII on the fly and sent to the terminal-end user as ASCII.

(XUPL) Xmodem Upload : This command is similar to XDOW, but applies to single file uploading. When performing an ASCII upload, the file sent must be an ASCII file. If the file is PET ASCII, send it using binary protocol. If the file to send is an ASCII text file, send it using ASCII protocol, and it will translated to PET ASCII on the fly, and stored on disk as PET ASCII. This is important to note. For Xmodem ASCII transfers, the files stored on disk should ALWAYS exist as PET ASCII files. The ASCII protocol is used to translate TO ASCII for downloading, so that the user gets ASCII if he wants it, and TO PET ASCII for uploading, so that the user's ASCII file he is uploading will be saved as a standard PET ASCII file on the BBS.

(DEL) Delete A File : This command replaces the DEL command in the old file section, plus adds one extra feature. This extra option is the ability to kill files straight from the physical directories, something that was lacking in the old file section. When using DEL on physical directories, you can set the priority in the SETUP file so that only you, the SYSOP, can use the command, level "s" users can use the command, or no one can use the command, in which case it is disabled. With DEL on physical directories, you can avoid having to use the DOS wedge in the editor section. By adding a directory number to the end of the DEL command, you can switch the current directory, in the same way as the aforementioned commands.

(F) Free Disk Space : This command merely prints the blocks free on any disk directory, in case a user wants to know. This command, as well as I and R below, also allow you to switch directories by adding a directory number to the command.

(I) Information File : One of the features of a Punter BBS system, among others, is the ability to have long directory listings that contain all sorts of information for each file in a directory, along with a short one line description. When you have something like that, then you are dealing with large records, which mean slower access, harder maintenance, and large relative file size. The approach taken with the I command is to store only important information for each directory in a SEQ file, and leave the non-descrip files alone. Each directory can have it's own description file, or you may disable an information file for any directory using the SETUP file. You are free to put anything you want in an information file, in any style. It's nothing more than a bulletin that appears in the file section. If you only want one information file to describe all your directories, disable all other information files in the other directories, and use just one information file on one directory of your choice. When an information file is inactive, the request for information on that directory will be ignored, as if the command never existed. Information files exist in 2 file name formats, one for physical directories, and one for relative directories. "+r.x.inf" is the format for relative directories, and "+p.x.inf" for physical directories, where "x" is the directory number from 0-7. For example, the information file for relative directory 3 would be "+r.3.inf".

Module #008 : File Section Xpander (P7)

(R) Read A File : If you are a member of CompuServe, you may be aware of the fact that you can read a file when browsing a data library. The R command allows you to read a file from any directory, and type does not matter. You can read a program file, as well as a sequential or user file. You cannot buffer a program file with the R command and save the buffer, run it, and expect it to work. The reason being is that null bytes (0's) are converted to spaces (chr$(32)) when any file is read throught the BBS.

(UD) Upload / Download Status : This is the same command as the UD command in the old file section. See the BBS V3.0 manual for details.

User Accessibility.

With the new commands in the module being added, a way had to be designed to allow the user access bits in the user's record to work with these new commands. The following table summarizes user accesses for the new commands. Refer to the BBS V3.0 manual for user access definitions.

New Command	Directory Format	Old Section Command Access Equivalence
PRO	R/P	None/C,D,U
BBS	-/-	None/None
T	-/P	-/C,D,U
UL	R/P	SETUP file sets access
DL	R/P	SETUP file sets access
I	R/P	+*DOW/C,D,U
R	R/P	+*DOW/C,D,U
F	R/P	*None/C,D,U
UD	-/-	UD/UD
D	R/P	*None/C,D,U
DOW	R/P	+*DOW/D
UPL	R/P	+*None/U
XDOW	R/P	+*DOW/D
XUPL	R/P	+*None/U
DIR	R/P	*None/C
DEL	R/P	*DEL/SETUP file sets access
MD	R/P	+*MD/C,D,U,MD
MU	-/P	+*-/C,D,U,MU

As an example, MD shows that it exists both in physical (P) and relative (R) formats, the relative MD format requires access to the MD command, set in the user access byte, whereas the MD physical command requires MD access AND requires physical directory access, which is denoted above by the commands C,D,U. C,D,U are the standard physical directory access commands. Where none is mentioned, this means that only standard file section access is required. Where "-" appears, it means that there is no analogy for the directory format (ie. MU has no relative format version). Where "*" appears next to the equivalence column, this means that the command requires additional access based on the individual access to each directory. For example, if P DIR 2 is restricted, all commands relating to it are as well. Where a "+" appears, this indicates a command based on upload/download restriction times (See next page). A few commands have access set in the SETUP file for this module. This will be explained later.

Module #008 : File Section Xpander (PQ)

File Transfer Restricted Time Periods.

Each directory, physical or relative, can have it's own restricted time period. Uploading has a restricted time period, as well as downloading. The 2 are distinct. In the SETUP file, there are 64 bytes to set for time period restrictions. 32 are for uploading, 32 for downloading. Of the 32 for each, 16 are for the start time of the period, 16 are for the end time. Of the 16 start times and end times, 8 are for physical directories, 8 are for relative directories. This can be summarized below:

64 Time Values :	32 Upload :	16 Start Time :	8 Physical
			8 Relative
		16 Stop Time :	8 Physical
			Relative
	32 Download :	16 Start Time :	8 Physical
			8 Relative
		16 Stop Time :	8 Physical
			8 Relative

Each value represents an hour of the day. Thus, a value can be from 0 to 24. The numbers have the following meanings:

0	: Time restrictions disabled.
1-11	: 1 Am to 11 Am
12	: 12 Pm (Noon)
13-23	: 1 Pm To 11 Pm
24	: 12 Am (Midnight)

The start time and stop time have the following relationship:

-The ending hour represents the hour at which time the restricted period ends. The starting time represents the time at which the restricted period starts. For example, a start time of 11 and an end time or 15 will result in a restricted period from 11 Am UP TO 3 Pm (Time values are military: 1500h is 3 Pm). This means that at 3 Pm, the period ends. This does not work like TIMELOK, where 3pm would be an inclusive hour. 2:59.59 is the last second in which the restricted period exists.
-A start value or 0 AND a stop value of 0 disables the restricted time period.
-A start value equal to the end value (But NOT 0) cause the period to run a full non-stop 24 hours.

Examples: The numbers given are the start, then the stop values.

15-14 : 3 Pm - 2 Pm / 1-1 : 1 Am - 1 Am (24 hours) / 0-0 : Period disabled / 24-12 : Midnight - Noon
24-1 : Midnight - 1 Am / 1-23 : 1 Am - 11 Pm / 24-24 : Midnight - Midnight (24 Hours)

Module #008 : File Section Xpander (P9)

SETUP Procedures.

The SETUP for this module is quite different than that of other modules. There are 2 files to the Xpander module. The file OLINK.M008 is the boot file for the module. This is the file OverLink loads on the BBS, then executes. This file will check the user's file section access, and verify the message write/download feature (see below), and then proceed to load in the main module if the user has access to the file section. This main module is named either +-+M008 or +-+M008.A. The files LINK.M008.A and LINK.M008.C are the linking programs to create the modified modules for the file section. Both SETUP.M008 and LINK.M008.C are part of the setup procedures. The 2 module files +-+M008 and OLINK.M008 will go onto the OverLink disk on your BBS AFTER it is modified. In short form, this is the way you make the OverLink module for the expanded file section:

1. LOAD, LIST, Modify, then SAVE the SETUP file on your library disk backup copy. Make sure ALL the module #008 files are on drive 8:0. This is IMPORTANT. Both LINK files, OLINK files, +-+M008 files, and the SETUP file must be on unit 8:0.
2. RUN the modified SETUP file. During this step, two temporary data files are written to the disk that contain the custom data you set in the SETUP file. Once those 2 files are created, the LINK file will be loaded on top of the SETUP file, hence replacing it. It will be run, at which point it will load the OLINK and +-+M008 files into memory. Then those 2 files on disk are removed, and the files in memory are merged with the data files on disk to form the new modules.
3. The data files will be scratched from the disk. During this process, the following willl be displayed to let you know that the process is operating correctly:

```
Linking File 1...+
Linking File 2...+
Removing Data Files...
Done!
```

4. You may now transfer the new OLINK and +-+M008 files to your OverLink drive. If you are running ASCII, then the modules are named +-+M008.A and OLINK.M008.A. Otherwise, they are as above, without the .A suffix. The OLINK file must be renamed to the OverLink command word you are using (ie. for "FSECT", name OLINK.M008 as "+-FSECT". The +-+M008 file must NOT be renamed. Put it on the disk as it is named, so that the secondary load will find the right file. If you are not using the CWI, then you will have to assign the correct module number to it (ie. "+gc.15" for CBBS O-L module #15). You do NOT have to put the LINK or SETUP files on your O-L drive. Those are part of the setup procedures only.

The next page(s) will describe how to go about modifying the SETUP file itself. It is a fairly lengthy process, similar to setting up the CREATE.CNF file.

Module #008 : File Section Xpander (P10)

The SETUP File : Line numbers are given next to each parameter to change.

1070-1330. These are the 22 prompt strings for this module. String lengths can only be up to 255 characters, so be aware that you can't concatenate long paragraphs into one string. In the case of prompt string A2$, the protocol select string could not be represented in one physical line number, so it was necessary to concatenate A2$ a few times. If for some reason, your string length were to exceed 255, then you will have to modify the program to write out the extra string variables that hold the extra length. For the modest BASIC programmer, this shouldn't be too difficult. Remember that B4$-C1$ are the 8 PHYSICAL directory descriptions. Make a trial run of the file xpander, until you are sure of where each prompt string appears, before you make any drastic modifications to the text strings.

1350. Version Of The Module To Modify : Set to 0 for color, 1 for ascii. Color uses the files OLINK.M008, LINK.M008.C, and +-+M008. The ascii version use OLINK.M008.A, LINK.M008.A, and +-+M008.A.

1360. Protocol Default : This is the default Xmodem protocol that will be enabled whenever the user enters the module. It may be change with PRO at any time. Set to:

0: SUM, Binary   1: CRC, Binary   128: SUM, ASCII   129: CRC, ASCII

1370. Carrier Signal Status : Set to 0 for normal carrier detect logic, set to 1 for inverted carrier detect. See The intro to BBS V3.1 at the start of this manual for an explanation.

1380. Display Directory Flag : Set to 0 for the current directory number to show next to the main file command prompt. Set to 1 to hide the current directory number.

1400-1570. These are the 18 command definitions for the new file section. They must be 4 characters long, padded with blanks if necessary. These definitions take priority over the file commands in CREATE.COM, unless you access the old file section, where the old command structure still exists.

1590-1600. Command Lengths. The command strings you make above must have their lengths set here. This differs from the CREATE.COM file, where the command strings and their lengths were put side by side. Make sure you align the length bytes with the strings. For example, the 5th value in the first set of DATA statements is a 2, which is the length for the 5th command, which is set as "DL ". For some commands, like T, that you don't want to use, set the length byte to a value of 0.

1620-1630. I Command Display Flags. For the information (I) command, you can enable or disable the file display for any of the 16 directories. The first set of 8 "Y"s is for the relative directories, the second set is for the physical directories. If you want the I command to be active, use a "y" for that directory, otherwise use an "n". For example, "ynynynyn" enables I for directories 0,2,4, and 6, but not for directories 1,3,5, and 7.

1650-1660/1680-1690/1710-1720. Border/Background/Text Colors : 11 of the new file section commands use their own colors accessed through this small color table. The old commands, which have been combined into the new section, will still access the old color table. See CREATE.COL for details on color table setup. This table is exactly the same, except that there are only 11 values. The command correspondence is:

1: PRO	2: BBS	3: T	4: UL	5: DL	6: I
7: R	8: F	9: D	10: XDOW	11: XUPL	

1740. DEL (Physical) Command Status : Set to 0 for level "s" and SYSOP access, set to 1 for SYSOP ONLY access, set to 2 to disable DEL from deleting files on physical directories.

1750. Upload Log Status : Set to 0 for general user access, set to 1 for level "s" only access, set to 2 to disable the upload log entirely.

1760. Download Log Status : Same as upload status above, but applies to download log.

Module #008 : File Section Xpander (P11)

The SETUP File : Line numbers are given next to each parameter to change.

1780-1790. Upload Log/Download Log Drives : The first value in line 1780 is the device that the upload log resides on, the second is for the download log. Line 1790 is the drive numbers for the upload and download logs, respectively.

1810-1830. Bank #2 Drive Table : This is the drive setup table for physical directories, and is toggled in with the T command. The drive numbers are assigned individually. There are 8 device numbers and 8 drive numbers, 1 pair for each of the 8 directories.

1850-1900. Information (I) Command Drive Table : The I command gets it's information files from the drives pointed to by this drive table. The first 8 device numbers and drive numbers are for the relative information files (+r.X.inf) and the remaining 8 device and drive numbers are for the physical directory information files (+p.X.inf).

1920-1990. Restriction Time Period Table : These 64 bytes were explained in detail 2 pages previously. The order of the DATA statements is organized as follows:

```
1920 : Upload   Time Restriction - Relative Directories 0-7 - Start Time
1930 : Upload   Time Restriction - Relative Directories 0-7 - End Time
1940 : Upload   Time Restriction - Physical Directories 0-7 - Start Time
1950 : Upload   Time Restriction - Physical Directories 0-7 - End Time
1960 : Download Time Restriction - Relative Directories 0-7 - Start Time
1970 : Download Time restriction - Relative Directories 0-7 - End Time
1980 : Download Time Restriction - Physical Directories 0-7 - Start Time
1990 : Download Time Restriction - Physical Directories 0-7 - End Time
```

The next few setup parameters are for the small OLINK boot module. This is part of the second data file that is created for the linking process.

2040-2050. These are the 2 text prompts for the small boot module.

2090. Master Message Write Toggle : Set to 0 to disable the ALL-WRITE process, set to 1 to enable it.
2100-2190. User ID# Table For Message Write/Download Process.

Both lines 2090 and 2100-2190 set up a procedure that will require users to write a message in the message section before being allowed access to the file section. This was implemented as an extreme measure to take against certain users who have compulsive downloading habits. The BBS will check to see if any user is in the ID# table you create, and if found, he will be told to go to the message section and write a message; otherwise, he may not access the file section. The master toggle in line 2100 requires that EVERY user write a message before they can go to the file section. This is a drastic option to get your message section going. This master toggle takes priority over the ID# table, no matter what you have it set to. The ID# table contains a list of from 0-100 user ID# of those users that HAVE TO write a message. If you find that you have more than 100 of these "bad" users, you will have to use the master toggle, or else resort to other means. Note that the SYSOP and level "s" users are exempt from this option no matter what you set for the master toggle or ID# table. That is why ID#0 in the table is a null ID#. Just key in all ID#s you want for forced message writing. They can appear anywhere in the table.

After this table, there is no more parameter data to set. Save this custom file AS any file name, then RUN it. You may modify the file anyway you want. Note that the LINK.M008.x files expect to use drive 8:0 at all times. Don't try to use 2 drives for this module or a drive other than 8:0, unless you want to disassemble the LINK files.

Module #009 ; Alternate TRIVIA Section (P1)
Files ; SETUP.M009 - OLINK.M009 - OLINK.M009.A

This is the alternate module for trivia, as opposed to module #002. This is more of a "true" trivia section, in that all questions are answered and verified as to whether they are correct at the time the user enters his answer. The basic structure of this section follows module #002 exactly, except for the mode by which questions are answered.

There can be from 1 to 25 sub-sections, and if there is only 1 sub-section, it will be treated as a single section only, without a master menu for accessing multiple sub-sections. The master menu is named "+u/menu", as opposed to "+t/menu" that module #002 uses. The outline is the same: set it up as a description of all sub-sections, as you would do with the master menu in module #002. If you have only one section, then this file is not needed, and the user would proceed to section 1 without having to make a selection. Entering "?" will re-display the "+u/menu" file, entering RETURN will exit to the main BBS.

Once in a section, the intro file is displayed the same as would be in module #002. You should create this file so that is describes the current trivia section. This file is named "+ui/xx", where "xx" is the section number. For example, the intro for section 9 is "+ui/09".

The user will be asked if he wants to continue and attempt to answer the questions. If so, the user's name and ID# are written to the result file (named "+ur/xx", where "xx" is the section number), as is the case with module #002. If the user responds no, then he will be placed in the master menu again, or back to the main BBS if you only have 1 section. If the user is a level "s" user and responds with a no, then he will be asked if he wants to see the results for this section, and then asked if he wants to clear the result file.

The question file is named "+uq/xx", where "xx" is the section number. This question file is where the difference in trivia structure comes into focus. You have to be VERY exact in creating this file, or the trivia module will go off into neverneverland. The format for this file is:

-Question Data, With Answer Choices
-An "@" delimiter to indicate the end of the question.
-The number of choices that you have in the multiple choice selection.
-The number that represents the correct answer.
-The text data that will be displayed if the user is correct.
-An "@" delimiter to indicate the end of the above data for a correct answer.
-The text data that will be displayed if the user is wrong.
-An "@" delimiter to indicate the end of the above data for a wrong answer.

-The start of the next question, or end of file text + the "&" delimiter to indicate the end of the file.

The above 8 parts make up one question block. Here is an example:

```
What color is not part of the basic visible light spectrum:

1) Blue    2) Red    3) Violet
4) Yellow  5) Pink@  <-------- No Carriage Return.
55That Is Right!@That Is Wrong@
```

The above show the question with possible answers, the "@" delimiter, the number of choices (5) and the right answer (5). This is followed by the text for the right answer, the "@" delimiter, the text for the wrong answer, and the "@" delimiter. This is one question block. The end of file text can be simply an "&", or can be a final comment ended by an "&". Otherwise, you make up the next question block.

Module #009 : Alternate TRIVIA Section (P2)

The limit on the number of questions you can have per section is 256. The answer file will be much smaller than that of the first trivia section because only 1 character answers are stored. The basic format of the question file is multiple choice. Answers can be any digit from 1 to 9. You specify the number of possible answers for each question as you go along. This number is stored as part of the question file to make things easy. The answer is stored right after that. The idea of having text displayed after a right or wrong answer was added for emphasis on the questions. You can make sly remarks about wrong answers, or casual remarks about the right answer. If you do not want remarks for the right or wrong answer, then you must place at least "null" equivalent character in the proper text area. You could add just a blank space instead of a comment. This will not affect output if you do this.

Note that the module itself displays the prompt that asks for an answer to be input; you need not enter this in every question. Also, the module displays a prompt for the right answer and the wrong answer. This prompt is the same each time a question is answered. This is why additional comments can be added to the question file.

As users answer questions, the results are stored in memory. When a user chooses to quit (entering "x" at the choice prompt does this, or the last question will too), the results are saved to the answer file. The user will then be given his results: the number of questions answered, and the number he answered correctly. The results stored in the answer file would look like this:

```
User ID# - Name
Y122Y2Y3Y2YYYY2Y21Y
(10/19)
```

The first line is obvious. The second line is the actual answers stored in order. Leftmost is the answer to question 1, rightmost the answer to question 19. A "Y" in any position indicates that the user got the question right. A digit in any position is a wrong answer. This digit is the user's choice for what he thought was the right answer. Since it is not a "Y", you know it is a wrong answer, and this digit represents it.

The last line is the number of questions correct / the number of questions answered). As you can see, this occupies much less memory than the other trivia module. However, you are restricted to a multiple choice system. Ten free blocks must be available for a trivia section to be accessed.

The SETUP File.

100. Set V=0 for color, set to 1 for ascii.
120. DF$: Destination file name.
140-245. The 12 text prompts for this module.
350. Change this line to modify the destination drive.
5000. Connect Time Flag : Set to 0 for connect time to stop during module access, set to 1 for normal run time.
5005. Number Of Trivia Sections : A number from 1 to 25.
5010-5020. Drive for the "+u/menu" file.
5030-5080. Drive table for the 25 trivia sections. See module #002.

Module #010 : On-Line Ordering Section (P1)
Files : SETUP.M010 - OLINK.M010 - OLINK.M010.A - CREATE.ORD - ORDER.RUN - ORDER.DLM

This module allows you to run a system with the capability of operating very simple "on-line shopping", similar to one of the services (stores) of CompuServe's Electronic Mall. Although most of the users of this BBS will not use this module, it is still here for the few who might have a business where things may be sold using the BBS as another outlet for product purchasing. This module is fairly lengthy, therefore the steps for setting up this section will be given in the order in which you should follow.

A. Using CREATE.ORD. This program is used to create, modify, and maintain your item files. You may have up to 25 sub-sections, the standard number for multi-section modules. Item files are relative files, each file containing from 1 to 195 items. The module sets the file to a fixed record count of 195, which amounts to 33 blocks per item file. Item files are named "+or.xx", where xx is the section number. For example, "+or.08" is the item file for section number 8. If you are using a single section module, the item file section number defaults to 1. These realtive files must be created with the CREATE.ORD program. Creating a new module erases the items and file of any previous item file you may have created. For the first time setup, you must create the item files for each section.

The record structure for each item is as follows:

Bytes 0-34 : Item Description. Each item can have from 1 to 35 characters for its description. This description is shown as the main title for the item when the user browses the items for each section while on-line. Thus, the standard description would be the item name only.
Byte 35 : Item File Number. The item file number represents the item description file number (See Below).
Byte 36 : Item File Position. This is the position value within the item description file.

Bytes 35 and 36 work together to specify where the BBS is to find the description text for each item in the item file. Description files are completely managed by you. They are simple SEQ text files that contain a short paragraph or two about the item. This description should contain the price and title of the item. If you run the color BBS, do NOT use special characters like cursor values and clear screens. Try to keep the size as small as possible if you intend to link descriptions.

Descriptions can be linked in the same file by separating each one by the "@" delimiter, the same character used in other modules. The item file position byte (36) is a number from 1 up to 99. An example linked description file would be like this:

```
(Text for item #m)@
(Text for item #n)@
(Text for item #o)@
```

Byte 35 is the description file number. This number can be a value from 1 to 195. Description files have the format "+des.xx/yyy", where xx is the module section number, and yyy is the description file number mentioned above. For example, description file 34, section 23, would be named "+des.23/034". As was said before, you are responsible for maintaining these text files.

In the example above, the "@" character must appear at the end of each description. Even if you are only using one description per file, you must end it with "@", since it is also used as an end-of-file marker, in addition to a delimiter. Item m,n, and o would have position numbers of 1,2, and 3, respectively. This is the position value in the record (byte 36). So, if byte 35 has a value of 34, and byte 36 has a value of 3, then this would correspond to the above example file name, and item #o in the above description text. The values m,n, and o can have any numerical value. For example, the items 11,189, and 134 could be assigned to m,n, and o respectively. The numerical assignments are left up to you.

Module #010 : On-Line Ordering Section (P2)

Bytes 37-39 : Item Price. Prices are stored in this record as a 24 bit binary value. This allows for a price range from 0 to 16,777,215. Because of the necessity for dollars and cents values, the price has a fixed decimal point 2 positions to the right. Therefore, prices run from $0.00 to $167,772.15. This should be quite adequate for your ordering section, provided you do not use it for dealing with real estate. Since this represents the maximum value for pricing, the total cost of a user's order cannot exceed the maximum value without causing a price overflow.

Bytes are stored in low, high, extended high byte order, where extended high byte is the value DIV 65536. The value MOD 65536 is left for the standard low and high bytes.

An added carraige return to the record makes a 41 byte record. CREATE.ORD maintains these records for you. The standard way of interfacing the text description files with the item relative files is to modify the item records when changing the structure of your items. For example, if you have a description file with descriptions for 5 items, and you decide to remove item 3 from this text file, you do not have to remove the text description from within the text file if you wish to add a new item. It is easier to change the item file numbers and position values in the item records to point to new description files than it is to move blocks of text in a text file. Always set up the relative files before making the description files.

CREATE.ORD Options. When you first run CREATE.ORD, you will not have a data file for your ordering section. This data file, named "+o/data", contains information for the ordering module to use during the BBS-User interaction. At any time the program requests the disk for the data file, you may enter F1 to create a new data file. The values that are placed in the new data file depend on what they are currently in memory. There is an option in the program to modify the data file's parameters, explained later. Once a data file has been created or loaded, you will be given the main menu, with the following options (NOTE: CREATE.ORD may be loaded from ANY drive 0 device you want):

CHANGE SECTION, CREATE ITEM FILE, LOAD ITEMS, SAVE ITEMS, SORT ITEMS, ADD ITEMS, DELETE ITEMS, MODIFY ITEMS, ITEMS SUMMARY, PAGED ITEM LIST, PRINT ITEMS, CHANGE DATA FILE, QUIT.

(1) Change Section. Before you switch to a new item section, you should always save your current section items to disk. The program only works with one section at a time, memory wise. Switching to a new section clears out the items of the old section. Section numbers range from 1 to 25.

(2) Create Item File. This option will create an item relative file for that section. If you already have an item file for this section, it will be replaced with the new one, and all items in the old item file will be lost. Thus, to clear an item file quickly, just re-create it. Note the following rules for file placement:

a) The ordering data file and the module data file (+ord/inf and +o/data) must reside on the same disk.

b) For each section, the description files, and the item file must reside on the same disk drive.

(3) Load Items. When you switch to a new section, or run CREATE.ORD the first time, the item count will be set but the item data (records) will not be in memory. Before you work with any section that has at least one item, you MUST load the items into memory. Otherwise, you will be working with garbage memory.

(4) Save Items. Any time you wish to reset the "+o/data" file, even when you don't want to save items, you should use this option. Saving the items will place all the records in memory in the item file, AND will also cause the data file to be updated. Always use this option before you switch sections, provided that the section you switch from has at least one item.

Module #010 : On-Line Ordering Section (P3)

(5) Sort Items. Sorting items will shift the item numbers, but will NOT change the item description file numbers and positions (bytes 35 and 36 in the record). This is why bytes 35 and 36 are there: to allow you to have your items in alphabetical order without worrying about having your description text files in the same order. When you add a new item to a a file, you simple assign it a new position and file number that you haven't used yet. The sort algorithm is a standard bubble sort. You may notice a slight delay if you have 195 items to sort, but still, you won't have to wait an eternity.

(6) Add An Item. Add, Delete, and Modify options will "loop". You will use that option over and over until you enter a RETURN to exit back to the main menu. When adding an item, the item number will be displayed, and you will be required to enter the following input:

DES:Item Description, 1-35 characters
ITF:Item File Number, a value from 1-195 (ie. 7 will be for file "+des.xx/007", where xx is the section number.
POS:Item File Position, a number from 1-99 (see NOTE below).
PRC:Item Unit Price, a value from $0.00 to $167,772.15. You must always add the cents value (ie. .XX, as in $.88).

You will be asked to confirm the addition at the end. NOTE: the ordering module must do a sequential search for the description text for an item that is not the first item. Thus, to display the text for an item in the 5th position in the description file, the BBS must read in and skip all characters that make up the first 4 descriptions. For example, if your first four descriptions contain 500 characters each, the BBS will have to read in 2000 characters before it finds the text for the item in position 5. On a 1541, this will take about 6 seconds. If you have 95 items in one description file, the BBS would have to read in 94*500=47000 characters, which would take about 3 minutes. Obviously, you must use as few descriptions per text file as possible in order to save time. Five is a good number. The reason why you should link descriptions in one file is so that you will save file space on your directory. If you use 1 description per text file, and you have 195 items, that means that you will use 195 directory entries, more than a 1541 can handle. If you use 5 descriptions per text file, you will only take up 39 directory entries. IEEE drives like an SFD are capable of handling probably 25-30 descriptions per file, thus occupying maybe only 7 files on disk. Of course, this all depends on just how much text you put into each description.

(7) Delete An Item. You are allowed to specify the item's full description name, or the item number (that is, its position in memory), given when you use the Summary option. NOTE: All items MUST have a description of at least 4 characters in length.

(8) Modify An Item. When you modify an item, you will be given the current value of the field in the record. If there is a field you do not want to change (ie. price), then just enter RETURN by itself, and you will skip to the next field in the record. Enter RETURN at all input prompts to abort the modification you may have erroneously started.

(9) Item Summary. This option will display the data for all items, as you entered them, starting at item #1. The listing will pause at the end of each item. Hit RETURN to continue scrolling through the items. Enter F1 while the output is paused to abort to the main menu.

(0) List By Page. This option lists all items exactly as they would appear to the user when on-line with the ordering module. Items are displayed in pages of 15 entries. Enter X to abort the listing after any page. Enter RETURN to move to the next page.

(A) Print Items. Output is not formatted with this option. Set your printer to top-of-page. The output given, from left to right, is: Item #, Description, Item File Number, Position In The File, and the Price. Use STOP to abort the printout.

Module #010 : On-Line Ordering Section (P4)

(D) Change Data File. This option allows you to change the current parameters of the "+o/data" file stored in memory. You must use the Save Items or Create Item File options to update this data file on disk. In addition, using F1 to create a data file when you are given the option will also update the data file. The parameters you may change are as follows:

1. Billing Methods. When dealing with on-line ordering, options like cash and cheque are not really of value, since orders are usually pre-paid before they are shipped. Thus you have the other options: credit cards and C.O.D. You are only allowed to specify up to 4 methods of payment for the user to make. The methods are definable by you. Enter the "1" key to cycle the value from 1 to 4.

2. Billing Queries. When a user is prompted for billing information, you are required to tell the user just what you want to know. For each of the 1-4 billing methods you have, you may specify from 0-5 input responses that the user must make. For exanmple, if you use VISA as a billing method, then you might ask the user the following:

Card Number.
Expiry Date.
Issuing Bank.
Cardholder's Name.

The above 4 options are required input for the user. The actual text defining each response is left up to you by modifying the ORDER.DLM text prompt file. When you enter "2" for that option, you use the cursor up/down key to change the first number, and the cursor left/right key to change the second number. For the X:Y values shown, X is the billing method currenty selected. It runs from 1 to the number of billing methods set with option 1. Y is the number of user inputs you want. The above example would require a value of 4. Note that you can specify a value of 0. For example, C.O.D. usually does not require any information other then the user's name and address, and phone number. If you use a value of 0, the user will not be given any inputs. The upper limit is 5 for each method. You must change ORDER.DLM so that your text prompts correspond to what you want as input from the user. Enter RETURN when you are done with option 2.

3. Address Responses. The module will always ask for a billing address and a shipping address. If the user's shipping address is the same as his billing address, the user will not have to enter the information twice. A usual setup for address input is Name, Address, City, State/Prov, Postal Code, and Telephone Number. This would be 6 input responses. You are allowed from 1 to 10. Once again, you must change the ORDER.DLM file if you want your own text prompts. Enter "3" to cycle the number values.

4. This option sets whether a user has registered his billing information with you. Sometimes, a user will give his billing/shipping address information, and his billing method. This information may not change for a long time period, and if this user is a regular customer, you can set it up so the user does not have to go through the lengthy process of entering billing information every time he wants to order something. Enter "4" to select option 4. The first 3 digits are the user ID numbers. Note that the SYSOP (you) has a number as well, even though you may never use it. Use the cursor up key (SHIFT held down) to move DOWN through the user ID numbers, and use the cursor down key to move UP through the user ID numbers. There are 256 ID numbers, one for each user. You use the cursor left/right key to toggle the second value between a "y" and an "n", which indicate YES and NO. A "y" (YES) value means that the user can use the billing information you already have for him. The BBS will then always ask the user if he wants to use current billing information. He still has the option to use new information at any time.

In addition, the F1,F3 and F5 keys, entered from the main menu, will change the screen colors. Once you have set up all your item files, you will proceed to create the description files.

B. Creating Description Files. This requirement was already discussed earlier, so only a short summary is given. Description files are SEQ text files, with the format "+des.xx/yyy", where xx is the item section number, and yyy is the item file number (ITF in the CREATE.ORD program). Each description must end with the "@" character, even if you only have one description per file. Obviously, you may not use the "@" character in the actual body of the description text, since it is a delimiter. If you only use one item description per file, the position number for every item (POS in CREATE.ORD) will always be 1. If you want to link descriptions in the file, just add another description, ending with another "@" character. Position number limit is 99, but will usually only be about 5 for a 1541 drive, a little more for an IEEE drive. Try to avoid using excessive screen control characters (clear screen) in your description text if you run a color system. Always place your text for each section on the same drive as the corresponding item file for each section.

C. Modify ORDER.DLM. .DLM files are delimiter files. These files contain the "@" character as a deliemter character, used to separate text prompts. The BBS text file (BBS.TXT) has been converted to a .DLM file, so that you may make changes to it without having to use CREATE.TXT. Normally, text prompts on the BBS system are separated by a NULL character (value 0). NULL characters cannot be used as text, and there is no way to generate nulls on a text editor, since a null represents "no-key-pressed". Other than that, the text files the BBS uses for text prompts are standard SEQ text files that a word processor can handle. The program CONVERT.DLM on your OverLink disk will take these .DLM files and turn them into the standard text prompt program (PRG) files, replacing delimiters with nulls as part of the conversion process. (See CONVERT.DLM in this manual for details).

In the case of ORDER.DLM, you do not use CONVERT.DLM to create a proper text file. Instead, the SETUP.M010 file will do this for you. Nevertheless, the format for the text file is still .DLM. For reference, you should print out the ORDER.DLM file to see how it is set up, or else load it into your favorite word processor. There are 57 text prompts that make up this file. Size is very important. You must make sure that your text file does not exceed the limit such that when the module has been setup with SETUP.M010, the end address of the load does not pass $CC00 (52224), or else your text will overwrite the screen memory, and possible the C64's I/O memory block. The current size of ORDER.DLM leaves about 8 more blocks of room (about 2000 bytes), which should be plenty.

Text prompts are numbered from 0-56, although you won't see the prompt numbers as you would in TEXT.PAL. Before the start of each new prompt, you will see the character "@", followed by a carraige return. This character pair (@+<CR>) is the delimiter between each successive text prompt. When you run the SETUP file, ORDER.DLM will be read into memory, and the character pair (@+<CR>) will be removed, and replaced with the NULL (value 0) character.

.DLM files must always have this format. So, the chain would be @+<CR>, text prompt 0, @+<CR), text prompt 1, @+<CR>, text prompt N-1, @+<CR>,@@. Note the way a .DLM file ends. You must end the file with the four characters: @, RETURN, and then 2 more @ characters. If you do not do this, the computer will go into an endless loop looking for these four characters. Note that we ended with prompt N-1. Prompt values always start with base 0 instead of 1.

You may freely modify the text any way you want, then save it back as a SEQ test file, with the name ORDER.DLM. No other name is acceptable, since SETUP.M010 expects this file. You must make sure that you have the exact number of text prompts. Do not remove any. You will note that where we have indicated that none are in use, we put the word "Empty". You can use anything you want for a prompt that you won't use. Even a single RETURN character will suffice.

To help you find out where certain prompts are in the file, and what numerical value they represent, a summary of portions of the ORDER.DLM file is given.

Module #010 : On-Line Ordering Section (P6)

0: Section Number Input Prompt.
4,6: Ordering Information And Summary Headers.
7: This prompt can be anything you want (order excludes tax etc....).
9: Given If User Has Registered Billing Information (See CREATE.ORD).
10: Ordering Form Header (User will always be given these 3 options).
11-13: Billing Address, Shipping Address, And Billing Method Headers.
14-17: Billing Method Descriptions.
19-28: The 10 Prompts For Billing/Shipping Addresse.
5,29,30: Quantity, Description, and Price Prompts Displayed During Ordering Summary.
31: Total Order Cost.
32-36: The 5 Prompts For Billing Method #1.
37-41: The 5 "" " "" "" #2.
42-46: The 5 "" " "" "" #3.
47-51: The 5 "" " "" "" #4.
52: Displayed After All Is Said And Done. Users May Not Turn Back If They Say YES Here!
53: Prompt To Confirm Shipping Address Same As Billing Address.
54: BBS Return Prompt.
55: Displayed After User Order Saved To Disk.
56: Displayed Under Summary Of Shipping Address IF Same As Billing.
1: Displayed At The End Of Each Page Of Items. X Exits/Orders. # To Get Item Description. RETURN For Next Page.
2: Displayed At The End Of Item Description. O Puts Item In User's "Shopping Cart".
3: Displayed If User Tries To Overfill "Shopping Cart" Without Emptying First.
8: General IS-THIS-CORRECT, OK, ALRIGHT Prompt.

D. Using SETUP.M011. This file is a 2-stage setup module, as is the file expansion module. After the SETUP program creates the data file ORDER.DAT, the program ORDER.RUN is loaded in and executed. This small ML program will merge the OLINK.M010 or OLINK.M010.A module, the ORDER.DAT file, and the ORDER.DLM file together, to create a new OLINK.M010 or OLINK.M010.A file. Always use a backup of your OverLink disk for processes like this, because a power out during a module setup like this could leave you without any module to modify. The ORDER.DLM file will be converted to standard text prompt format during the merge process. If the computer resets during the SETUP run, you are missing a file.

The data to be modified is contained in the 5000 line range like the other modules. Type LIST 5000- for the module parameters.

5000. Version to modify. 0 for color, 1 for ASCII.
5010. Drive that the main menu ("+o/menu") resides on. Ignored if only one section used.
5020. Number of ordering sections, 1-25.
5030-50. Section device table (see previous module setup procedures for details).
5060-80. Section drive number table.
5090. Drive that the "+o/data" and "+ord/inf" files reside on.
5100. Connect time flag. Set to 0 if you want users' connect times to stop during their stay in the module. Set to one otherwise, for normal running time.

NOTE: The file "+ord/inf" is where the user's orders are stored when they make a purchase. Details will be given in the next section.

Module #010 : On-Line Ordering Section (P7)

E. Using The Ordering Module. Once you have run the setup file, you must put the OLINK file on your OverLink drive. It is the only file that you put on it. This is a large program, like the terminal link for the BBS. As such, this module writes into the storage area for the standard BBS text file. You must therefore place your BBS text file on your OverLink drive, and name it "+bbs/txt". When a user exits the ordering module, the text file will reload. If you have already done this by installing the terminal link, then you don't have to do it now.

You must create the "+ord/inf" file. This is a SEQ text file, and does nothing more than hold user's orders. You may initially place any data in the file you want (ie. 2 RETURNS, a header description, etc.). As orders come in, they will be appended to the this file. Place the file on the same disk drive that you put the "+o/data" file on, which you set in the SETUP data.

You are now ready to use the ordering section. When a user enters this section, the contents of the "+o/data" file are read into memory. Then, if you are running more then one section, the ordering section main menu is displayed. This file is named "+o/menu", and is analogous to the other master menu files in other modules (ie. +t/menu in trivia module), in that you put in text describing each sub-section. If you are using one section, the module sets the section value to 1, and the user proceeds to the items display. Otherwise, the user enters an item section number. However, at this point, the module checks to see if the user has placed any orders. If the user tries to hit return to exit the module (in case of more than one section), or the user attempts to exit by entering 'X' from any page in the item display (for one section), the module will see if an order for any items has been made. If so, the user will proceed to the ordering information section.

When the user enters a section, pages of items are displayed, 15 at a time, giving the item number, and it's description (the one you made with CREATE.ORD). After each page, a user may enter "X" to exit, a number that corresponds to an item number, or RETURN to go onto the next page. In addition, a level "s" user may enter "r" to read the current ordering result file (+ord/inf). If a user enters "x", he will either exit back to the main menu in the case of multiple sections, or he will be asked for ordering information IF he placed any orders. If not, the user will be returned to the main BBS. When selecting an item number, it does not have to be one that is on the current page. For example, if the user is on page 1 (items 1-15), and he enters 35 (for item #35), he will still get to see the description for item #35. After seeing the description, the user will be placed back at the current page, not at the one for item #35 (which would be page 3). The module remembers the current page number at all times, independant of item selection. A user may not select an item number of 0 or one greater than the number of items you have. After all pages have been displayed, the user will be returned to the previous menu.

When a user enters an item #, the module will search for the item's file number for that section, and when found, will initiate a search within the file for the position of it's description. If the position within the file is the first one, no search is necessary, and the description is immediately displayed. During a search, for each item description that has to be read past, a period (.) will be displayed to the user to let him know that a search is being made. Generally, for small descriptions (<half a page), this will go too fast for the user to notice.

After a description is displayed, the user will be given the chance to order this item. Entering "O" places the order. Hitting RETURN (or anything else) places the user back to the current item page. As orders are placed, they are stored in an item order list. This list has been limited to 25 items per order. This does not mean a user can only order 25 items. It simple means that a user must proceed to complete the order for the current items selected. After that, the use may order 25 more items.

Module #010 : On-Line Ordering Section (P8)

Placing The Order. If a user has ordered one or more items, the ordering sequence will begin when a user attempts to exit the module. The sequence of events are as follows:

A. Quantity Entry. Each item's description and price are given. The price is the one in the item record, not the item description file (they shoule be the same though!). The user must enter how many of the current item he wants. A user may order up to 99 of any item. The module keeps a running total of the user's order (like a cash register). Each item ordered will have to have a quantity input by the user. After all items are done, they are reviewed for him. During the output of ANY text in the module, the list control keys are active (S,C, and A for list control), so that the user may pause the review if it is going by too quickly. After all items are done, the total for the order is displayed. The user now has the option to confirm this. By not entering a quantity, a user will abort the ordering process, and his order will be cancelled.

B. Billing Address. The user must answer all questions you give for all remaining prompts, or the order will be cancelled. The billing address info is setup as previously mentioned. Up to 25 characters are allowed for input for each question.

C. Shipping Address. The user will have to enter more address info, unless of course he selects same as billing address. If so, this section is skipped.

D. Billing Method. The user must select one of your billing methods. When one is selected, he will then have to answer any questions you created (from 1-5). If you set the queries to 0 with CREATE.ORD, then this part is skipped too.

E. Information Summary. Each of the user's inputs from parts B,C, and D are displayed to the user in summary format. After each of the 3, the user may abort the order. If aborted, the user will start with part A again. He will not be returned to the main menu in this case. If all 3 inputs are verified okay, the user is asked to confirm the order. If the user responds yes at this point, then he can not cancel it. The order will be saved to "+ord.inf", and the user will be notified of this. Then the user will be placed into the start of the module, where he may order more items.

NOTE: If a user has registered billing information (set with CREATE.ORD), then the user will be asked if he wished to use his current billing information BEFORE the start of part B. If the user responds Yes, then parts B-E are skipped, and he will simply be aksed if he wants the order placed. If so, the prompt "*** Current Billing Info***" will be placed in the +ord/inf file instead of the actual ordering information.

Storage Format. What is placed in the +ord/inf file for each order is as follows:

User ID#-User Name-Method #   Example: 034-JOHN DOE M:1

Item Entries. For each item ordered, the following information is displayed: Section Number, Item Number, and Quantity. Example - S:13 I:144 Q:3. Item number is IMPORTANT! In this case, the item number is the actual position of the item in the current item file. Changing the item file at any time can change this number, so always process ALL current orders before you change an item file. Use the LIST BY PAGE option in CREATE.ORD to find out what the user has ordered. After the item information, the ordering information is displayed. Only the user input is stored, without the text prompts. You will have to match the input to your prompts to determine the user's input. Each of the 3 blocks of input is separated by a blank line. If a user selects current billing information, these 3 blocks of data will not be shown. Also, if you have no inputs for the billing method the user selected, that block (the 3rd one) will not be shown. The 3 input blocks, in order from top down, are Billing Address, Shipping Address, and Billing Method Information. You are resposible for clearing this ordering result file any way you want. The module does not include a clear result file option.

Module #011 : BlackJack
Files : SETUP.M011 - OLINK.M011 - +BJ.INST - CREATE.M011

This is the first of 3 simple game modules you will find on this disk. More will be added later, provided that OverLink is met with suitable response to allow the creation of additional OverLink disks.

The file "+bj.inst" is the instruction file. It is a color file that you should place on your BBS system as the module's help file. There are no ASCII versions of the 3 game modules, for reasons mentioned earlier. You should read the color help file to see if you think modification is necessary. All the instructions for playing the game are in this file. Therefore, we will skip the play mechanics, and just mention the setup procedures.

This module uses cursor plotting for the game dispaly, as does module #12. Because of this, DarkTerm 4.c must be used by the terminal end user. Plotting is built into V3.1 already.

BlackJack saves the user's current cash value and his high score to disk on a relative maintenance file. High scores represent the highest cash value the user ever reached up to his current session. The current scores represent just what the name implies. How much money they have currently. Only you can change the user's high score and/or current cash status. Both variables always update when a user completes a session.

SETUP file variables:

100. V : Version variable is ignored.
120. DF$: Destination file name.
140-190. These 6 prompts are written to disk as one prompt. Because of the size of this text prompt block, it was split into 6 variables, a0$-a5$.
200-210. The second prompt is split into variables b0$ and b1$.
320. Destination drive and device values.

5000. Connect Timer. Set to 0 for connect time stops, set to 1 for standard running time.

5010-20. Drive that the relative file "+bj/hi" goes on.

5030-40. Drive that the helpfile "+bj.inst" goes on.

5050. Number of decks of cards to use. You can use from 1 to 6 decks of cards at once. Casinos generally use at least 2 decks. Multiple decks will prevent card counting, used by cheaters.

5060. Shuffle marker. The marker is the card number where re-shuffling occurs. Casinos usually place the marker two-thirds of the way into the card stack. The module limits the card marker to values from 1 to 255. Do not use a marker value that exceeds the number of cards in the deck. Casino rules specify that top cards are not "burned".

5070. Dealer stand value. Usually the dealer will stand on 17. Some casinos use 16 for dealer stand. You can use any value from 1 to 21. Using a stand value above 21 will be somewhat unpredictable.

5080. Minimum bet. The bet values are dollars. The module allows bets to run from $1 up to $32,000. Try to use a narrow range to keep the users's cash value from jumping up and down like a yo-yo.

5090. Maximum bet. Make sure it's at least as large as the minimum.

5100-20. Border, background, and text colors. The values stored are the string values of the color to use. For example, white would be CTRL-2, in quotes.

Module #011 : BlackJack (P2)

Once you run the SETUP file and place the module on your OverLink drive, you must make up the user high score file. CREATE.M011 will create and maintain your high score file. But since it is just a high score file, their is no provision for in-depth record control that a relative file manager module would have. Load up and examine the CREATE file.

The first data value is the mode specifier (line 1010).

Value 0: The high score file will be created from scratch.
Value 1: The cash values for all 255 users are re-initialized. The high score values are not affected, however.
Value 2: The cash value and the high score value for one specific user are changed.

The two data values in line 1020 are the high score initial value, and the inital cash limit values for all users if mode 0 or mode 1 above is used. Cash values and high score values are signed 24 bit dollar values. This means that a user's cash value and high score value can be as high as $8,388,607. If a user's cash value goes above this, very strange results will occur, but will not be lethal to the BBS. High score values may not be negative values. The lowest high score a user can have is $0. However, the cash value can go as low as -$8,388,608. A user will have to make up his debts unless you re-initialize his starting cash value with mode 2.

It is suggested you make the high score and initial cash values the same when starting the high scores for the first time. The last three data values in line 1030 are for mode 2 change for a single user. The first value is the ID# of the user you want to change. The second and third values are the user's new high score and new current cash value, respectively. NOTE: You (ID# 0, SYSOP) do not have a value in the high score table. If you want to participate in the high score runs, you must add yourself in as a normal class user.

The CREATE program is small. Examine it closely to see just how the program works. Either run this on your disk that you want the high score table on, or use COPY.REL1 to copy the file to the drive you want it on.

Each record consists of only 7 bytes. The first three are for the high score, the next three for the user current cash value, and the last byte is a carraige return. Bytes are stored in the format : lowest 8 bits, middle 8 bits, high 8 bits.

Module #012 : BlackBox
Files : SETUP.M012 - OLINK.M012

This program was written to take advantage of the cursor plotting of the color BBS. There is no analogous program for the ASCII version. This game is a very simple one. The full instructions for this program are found on the OverLink disk, under the file name "+box.inst". This is the same file name that you will place on your system, as the help file for the users. Users of the system must use DarkTerm 4.C or later.

SETUP file:

120. DF$: Destination File Name.
230. Destination device and drive.

5000. Connect Timer. Set to 1 for time to run normal, 0 for user connect time to stop, while in this module.
5010-20. Drive that the file "+box.inst" goes on.

Note that whenever the cursor stops and sits at any point on the screen, a user is required to enter some input (ie. the number of atoms to use).

Module #013 : Polling Section
Files : SETUP.M013 - OLINK.M013 - OLINK.M013.A - CREATE.POL

If you have used the voting section on V3.0, you will be aware of the fact that each voting topic requires 258 bytes of storage. That means that 100 topics requires more than 100 blocks on disk. Also, there was no way to sub-divide the topics as is the case with the bulletin section. The polling module allows up to 10 sub-sections of up to 50 topics per section, for a limit of 500 topics. The section numbers run from 0 to 9, not 1 to 10. Remember this! As another option, a user may be able to write a poll message on the system. However, the module defers its addition to the polling topics by placing written topics into a poll storage file. This file must be broken down, and topics extracted, through the user of a wordprocessor. You must also add the description part of the user's topic to the description files for each section.

The section operation part of the module is the same as would be for the Trivia modules or the other multi-section modules, except for the fact that you only have from 1-10 sectios (not 25), and the section numbers run from 0-9, not 1-25. If you are using one section, you will be using section number 0 as the default. The master menu will be skipped in this case, and the user will not have to select a section number. If you have more than one section, then the file "+v/menu" will be displayed, and the user will have to enter a section number. This master menu file should be set up like other multi-section master menu files.

Once the user enters a sub-section, that section's description file is displayed. This description file contains the main titles for all of your topics for that sub-section. Because it is a standard SEQ text file, you may place anything in this file you want. Sub-section menu files are named "+po/menu.x", where x is the section number (from 0 to 9). For example, the description file for section 3 would be "+po/menu.3".

After the description file is displayed, the user will be able to : enter "?" to re-display the description file for that section, enter "w" to write a poll topic, enter "r" to read the topic file, or select a topic. To select a topic, the user enters the topic number he wants. If there is no associated topic data file, or the user enters an illegal topic number, the option will be aborted. This is similar to the way in which the voting section handles topics not yet implemented. If you have 15 topics in section 3, and the user types 16, the module would get a file not found error, and would abort the option, since the topic is not available.

Topic data files are SEQ text files that function in the exact same way as the voting section of the BBS. The file must contain a list of all possible voting selections. See the VOTE command for vote topic file organization. After the file is displayed, the user must select a topic value, or enter RETURN to not vote. If the user enters "?" instead of a voting value, the current results will be displayed to the user, based upon priviledged access settings for this module. The possible values are 1-9 for poll casting, which is the same as that for the voting section in the BBS.

Each of the sub-section value files (the relative files that contain the user's poll selection values (1-9)) occupies only 3 blocks on disk. This is because only 10 bytes are used to store the results for each topic, compared to the 258 bytes for the VOTE command. This dramatic drop in disk space has a price though. A user may vote on any topic as many times as he wants. There is no way the module can check if the user has voted at a previous time on the same topic. This allows for "fixing of the ballot box". It may just happen that users may illegally vote to change the proper outcome of a poll. Thus, you should realize that without honest users, you may end up with very inaccurate results. In this case, you should go back to the VOTE command.

The relative files that store the results of a poll are named "+poll/x", where x is the section number. For example, the result file for section 0 would be "+poll/0". The program CREATE.POL will create these result files for you. The text files that store the data and result outcomes for each topic are named "+po/y.xx", where y is the sub-section number, and xx is the topic number for that sub-section. For example, the topic file for topic #34, section 9, would be "+po/9.34".

Module #013 : Polling Section (P2)

The "r" option for reading the module topic file is only available to a level "s" user. When a user selects "w" to write a topic, all of his input is written to this file, or rather is appended. This file contains all topics written by all users from all sub-sections. You must maintain this file, extracting each topic individually with a text editor. This file is named "+poll/res", and is a SEQ text file that you must initially place on one of your BBS drives. It may contain any starting text you want to use. All topics written with "w" will be appended to this file.

The user may or may not be able to use "w" to write a topic. This option is left up to you. The level "s" user will always be able to write a topic, though. The standard line oriented text editor of the main BBS is used for writing topics. The range has been reduced from 100 to 50 lines, but the option to continue after saved still exists, so there should be no size problems. Each topic is stored as follows:

User ID#-User Name
Module Sub-Section User Was In At Time Of Writing:Description User Wants
Date Written

(Text Data)

Example:

034-JOHN DOE
1:This Is My Description

This is my text data.

If a level "s" user reads a topic, and sees the results of the poll, he will have the option to clear the results. Poll results are displayed as they are with VOTE. For each topic value that has been voted for at least one time, its value, followed by a colon, and the number of votes for that value are displayed (ie. 4:25 indicates that the vote value 4 for this topic has been voted for 25 times). The only way to clear individual topics is by selecting the clear option. Creating a result file clears out all topic results for a specific sub-section. A disk doctor may be used if you need to make drastic repairs to the results of a major portion of a result file.

Use CREATE.POL to create the topic result files. Line 1050, PN specifies the sub-section number for the poll file to create. Value range from 0-9. This is a fixed record size file. Each file contains room for 50 topics. Lines 1060 and 1070 specify the drive to write the file to. This is in case you want to create the files on an SFD, but have the CREATE.POL file stored on a 1541 or some other incompatible drive, incompatible with the destination drive for the relative files.

Using the SETUP file:

100. V: Version of module to change. 0 for color, 1 for ascii.
120. DF$: Destination file name.
140-220. These are the 9 text prompts for the module.
330. Destination file device and drive number.
5000. Connect time flag. Set to 0 for connect time to stop, 1 for normal running time.
5010-20. Device and drive for the BBS drive that "+v/menu" goes on.
5030. Number of polling sections (1-10).
5040-5070. Device and drive tables for the 10 sections.
5080. If set to 0, poll results for topics will be displayed. Set to 1 to restrict results to level "s" users.
5090. If set to 0, all users may write poll topics with the "w" option. Set to 1 for level "s" users only.

Module #14 : SuperWumpus
Files : OLINK.M014 - SETUP.M014 - WUMPUS.DLM - WUMPUS.RUN - +WUMPUS.INST

This game is an adaptation of the game SuperWumpus, written by Jack Emmerichs for 6800-based machines. Originally published in 1978, it has been re-written and modified to run on the CBBS. There is no ASCII version of this module, for reasons outlined earlier.

The instruction file "+wumpus.inst" contains all the information you need to play the game. This file must be placed on the system for users to access. This is another multi-stage link module, organized in much the same way as the on-line ordering section. The file "wumpus.dlm" is a delimiter-type file. For information on delimiter files, see the instructions in the on-line ordering section. This file contains 39 prompts that you may modify. The second prompt named "NO PROMPT" is not used by the module, but must contain some text anyways. You should familiarize yourself with the module before you begin to change any of the text. When using the setup file, the file WUMPUS.RUN, WUMPUS.DLM, and OLINK.M014 must alll be on the same disk as the setup file. The linking procedures follow the exact format as that of module #12. The setup parameters in lines 5000 onwards will now be explained.

5000-10. Device and drive for the help file "+wumpus.inst".

5020. Set this value to 0 for connect time to stop, 1 for normal running time.

5030. This value is the number of bottomless pits you want in the 20 caves. The value of 2 is optimum, but you may change this value. The sum of pits, bats, the wumpus, and you must never exceed 20, since there are only 20 caves.

5040. This is the number of caves that will contain the SuperBats. You must use a value of at least 1 for pits and bats.

5050-60. These 2 values specify the minimum number and maximum number of days of supplies the player will have for survival. You may set the 2 values as low as 1 or as high as 255. They may be equal to each other.

5070. This value is a binary divide factor that determines how much supplies a player will lose when he is picked up by superbats. The value of 2 means that 1/4 of the supplies will be lost. Other values that you may use are 1 for 1/2 supplies lost, 3 for 1/8 supplies lost, or 4 for 1/16 supplies lost.

5080. This is the number of arrows a player starts with. Values may range from 1 to 255.

5090. A value of 234 means that caves WILL BE scrambled at the start of each game. A value of 96 means that the maze of caves will remain the same from game to game. Any other value will crash the system. Remember, only 96 or 234. If you design a custom cave maze other than the dodecahedron, you should not scramble the caves, even though it is possible.

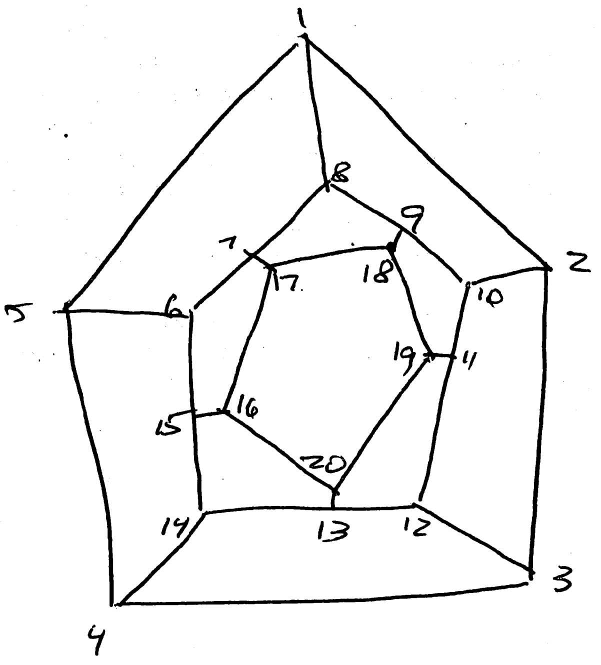
Lines 5130 to 5220 specify probability factors that determine whether certain events will occur during game play. All factors can have a probability from 1 percent to 100 percent by specifying values from 1 to 100. These events will be tested for occurence for every move the player makes during the game.

5130. Arrows shot will hit superbats.
5140. Arrow shot will fall into a pit (hint: tells where a pit is).
5150. Wumpus will move on next turn.
5160. Bats will migrate to a new cave.
5170. You find a throwing rock for which to kill the Wumpus.
5180. Rats invade the caves and eat half your supplies.
5190. Rock slide occurs and closes a tunnel.
5200. Player finds extra supplies.
5210. A rock thrown at a Wumpus will kill it.
5220. A shot arrow that goes wild will kill you if you're hit.

Module #14 : SuperWumpus (P2)

The 20 lines starting from 5270 are the connections for the 20 caves. Line 5270 represents the connections for cave 1, line 5280 for cave 2, etc., up to cave 20. For each cave, there are 3 tunnels that lead to other caves. The fourth value of each line in for each cave is always 0. The first 3 data statements on each line are the tunnels leading to other caves. When you make your own custom caves, you must make each tunnel connect to a cave number from 1 to 20. A cave CAN connect to itself. For example, cave 1 can have a tunnel lead to cave 1. In this case, you would be creating a dead-end tunnel, effective for making harder mazes. The normal cave pattern for this module is a squashed dodecahedron, which is a 12-sided cube. The graphical representation of the default maze is given below.

You are free to create any pattern you want. Just make sure all 20 caves can be reached by the player (ie. include all caves in your design). This game was released on the library disk after the official OverLink release. Please notify us of any problems with this module.



Miscellaneous Files (P1)

There are a few files on the OverLink disk that are not considered modules. Each will be explained, in the order they appear on the disk.

NOTICE. This program, when loaded ,8,1, gives the copyright notice for the OverLink.

YELLOW PAGES. This is a very handy public domain utility that will re-organize your disk directories. The program was not written by D.S.S. You should use it to re-organize your directories. A good idea would be to remove the "--" seperators from backup copies of your OverLink disk to make the directories shorter and faster. All instructions for that program can be found within the program. We recommend it as an excellent disk utility.

CWI.COM. This program is the one that makes up the "++xc" CWI file. The "++xc" file contains the command definitions for the CWI. The function of this file was explained at the beginning of this manual.

INSTALL DRIVES. This is the file that creates the DRIVE TABLE file, the one that is added to your disk #2 along with the standard V3.0 configuration files. Only "DRIVE TABLE" belongs on disk #2, not INSTALL DRIVES. The function of this file was outlined previously as well.

M.6VC.PAL	-PAL source code file for MODEM.6VC 1200.
M.TTEL.PAL	-PAL source code file for MODEM.TOTEL TEL.
M.1650SLOW.PAL	-PAL source code file for MODEM.SLOW 1650.
M.1670-2.PAL	-PAL source code file for MODEM.1670.2.

MODEM.6VC 1200	-Modem file for the 6VC 1200 Super Modem.
MODEM.TOTEL TEL	-Modem file for the Westridge/Total Telecommunications Modems.
MODEM.SLOW 1650	-Modem file for a Pocket Modem or 1650 compatible that has a slow response to answering calls.
MODEM.1670.2	-Modem file for the 1670, a new and perhaps(?) better version.

The source code was written with the ProLine PAL 64 assembler. With some changes, the source code will run on other assemblers, but we recommend this one. The object code files are your standard modem files, which you should have read about in the V3.0 BBS manual. The source code format follows that which was outlined in the Technical Notes section of the manual. If you find that your 1650/Pocket modem is answering the phone in strange ways with the old modem file, try the new one above. It introduces a slight delay after going on-hook before accepting incoming calls. If you find that your 1670 does not like to answer the phone frequently, try the above modem file. It sends the AT command sequence string twice, in case something garbled it the frist time it was sent. If you have a Hayes compatible modem, like the 6VC 1200, you may use the above 6VC modem file. Hayes modems use inverted carriers, and this file handles that properly. If your modem supports DTR, you can add a direct DTR disconnect to the hangup routine. For example, LDA #0:STA 56577 will disconnect the caller.

Miscellaneous Files (P2)

CONVERT.DLM - Delimiter file converter.
BBS-TXT.DLM - BBS V3.0 text file, in delimiter (.DLM) format.

Delimiter files, mentioned under Module #10, will be explained once more. The file BBS-TXT.DLM is, in essence, the BBS text file, which is named BBS.TXT on disk #2. The difference is that the BBS-TXT.DLM file has been put in a format that you can use with any text editor or word processor. The normal sequence of text prompts for a normal text prompt file, such as BBS.TXT, would be:

```
NULL
Text Prompt 0
NULL
Text Prompt 1
NULL
....
Text Prompt N
NULL,NULL,NULL
```

The NULL is a null byte, which is value 0. These zero bytes are used to separate prompts on the BBS. Each prompt starts with a NULL, and ends with the NULL of the next prompt, as you can see above. The very end of the file, after N prompts, would contain 3 NULLS, which tells the BBS that this is the end of the text prompt file. NULLS have no real value as input as the letter "a" would have, for example. Therefore, it is impossible to put these nulls in the file with any text editor. Delimiter files do not use nulls. Instead, the characters "@"+CARRAIGE RETURN make up the text prompt separator. The above sequence would thus be, in delimiter format:

```
@(RETURN)
Text Prompt 0
@(RETURN)
Text Prompt 1
@(RETURN)
Text Prompt 2
@(RETURN)@@
```

Note the ending of the text file: @+RETURN+@+@. These MUST always be the last 4 characters of the text file. To see a proper .DLM file, load up the BBS-TXT.DLM file in any text editor. Note that the files you make do not have to be named .DLM like a .ARC file with ARC. It's just there for notation. By using this format, you can bypass the CREATE.TXT program on disk #2. Also, you may add color to your text prompts without having to use TEXT.PAL and an assembler. CONVERT.DLM will convert your delimiter files to a more standard text file that the BBS can use. It will replace the @(RETURN) character pair with the NULLS it needs. Your delimiter file will be replaced with the proper file when it is run, so always use a BACKUP of your delimiter file with CONVERT.DLM.

Load Adress: 39424

Miscellaneous Files (P3)

RELATIVE COPY - This program is an update to the file COPY.REL1 on disk #2 of the BBS. This version only differs from the previous version in that you may specify a wider range of device numbers for copying the files.

CHARSET CONVERT - This small BASIC/ML program allows you to manipulate the character sets that the DarkTerm series of terminal programs and the BBS use as character sets. These include the 6 character set files on disk #2 (CHARSET x). Any character sets you've used on DarkTerm Versions 2.0, 2.1, 2.2, 3.0, 4.x (x=0,A,B,C,X) will work with this program. Using this program means that you do NOT have to use a machine language monitor to merge the character sets, or a disk doctor to set the load address to $E000 (57344). This program will convert character sets to DSS Terminal/BBS format or will convert DSS Terminal/BBS format files to normal character set format, in two 9 block files. Complete instructions are provided within the program. They will not be repeated here.

+HE.2 - This is a sample help file for the new expanded file section. Use it as a replacement for your old +HE.2 file, or use it as a reference when writing your own.

NOTE: Any late changes made to the OverLink disk (ie. late bugs, or added modules) will appear after this last page. These changes may or may not be documented in the table of contents. If you have any questions regarding this OverLink system, or the V3.1 system, don't hesitate to write us, or call our BBS Hotline ((416)-445-6788). As of this writing, we are currently working on a special high-speed version of the BBS that will run exclusively on the Xetec Lt. Kernal 20 megabyte hard disk drive. Stay tuned for further news.

CLOSING COMMENTS

Well This Wraps Up Our First Expansion Library Disk!!! This Should Keep You Busy For a Little While!!!

Development is UnderWay to Support the Xetec Lt Kernal 20 MegaByte Hard Drive! Once Done, We Will Tackle The ICT DataChief 20 MegaByte Hard Drive!!!

We Will ReDesign the Message Base, Incorporating NetWorking and AutoMessage Forwarding Between 1200 Baud Boards in the Near Future! HopeFully on OverLink Library #2!!!

Also, One Final Note to All Programmers, If You Have Written Any Modem Files For the BBS or Term, Please Send Them Along So That We Can Make Them Available to the Others!
All the Best in '87!!!
Andrew Leaver
President, D.S.S.