7100-02821

February 18, 1983

Included with this letter you will find some information to supplement the manuals supplied with your Corvus Concept Workstation. This includes the following:

- Miscellaneous Notes
- Upgrading the Corvus Concept Operating System
- Converting Old Character Set Files
- Converting Old LogiCalc Files
- Pascal and Fortran Enhancements
- Mixing Constellation II and Constellation I Systems
- Installing an Add-on Drive to an Existing Disk System
- Corvus Concept Display Driver ERS
- New Unitstatus Functions of the Keyboard Driver
- Using EXEC Files and Command Lines
- Two Examples of Startup Files
- Data Communication Parameters for Printers Commonly Used with the Corvus Concept
- Using the Linker and Library Utilities
- Program Space Requirements
- Building Action Tables and Alternate Character Translation Tables

In closing, we suggest that you start by reviewing this document then read the following manuals in the following order:

- <u>The Corvus Concept Personal Workstation Installation Guide</u>
- <u>The Corvus Concept Diskette Installation Guide</u>
- <u>The Corvus Concept Disk Installation Guide</u>
- <u>The Corvus Concept System Manager's Guide</u>
- <u>The Corvus Concept Personal Workstation User Guide</u>

NOTES ON THE DISKETTE DRIVE

DO NOT LEAVE DISKETTES IN YOUR DISKETTE DRIVE WHILE THE CONCEPT IS BEING POWERED-OFF OR POWERED-ON.

When following the instructions in The Corvus Concept Concept Diskette Installation Guide, connect the flat cable by matching the arrows on the connectors as depicted in the diagrams. The connection is correct when the arrows line up, even if the flat cable goes in a different direction than in the diagram.

NOTE ON THE EDWORD USER GUIDE

In Chapter 10 of "The Corvus Concept EdWord User Guide" under the heading "Making a Pattern," instructions are given indicating that a backslash (\) must precede the letters designating each field. Though the illustrations in that section omit the backslash it is important not to omit the backslash before each field designation in your pattern, as the Forms command will not operate without the backslash preceding each field designation.

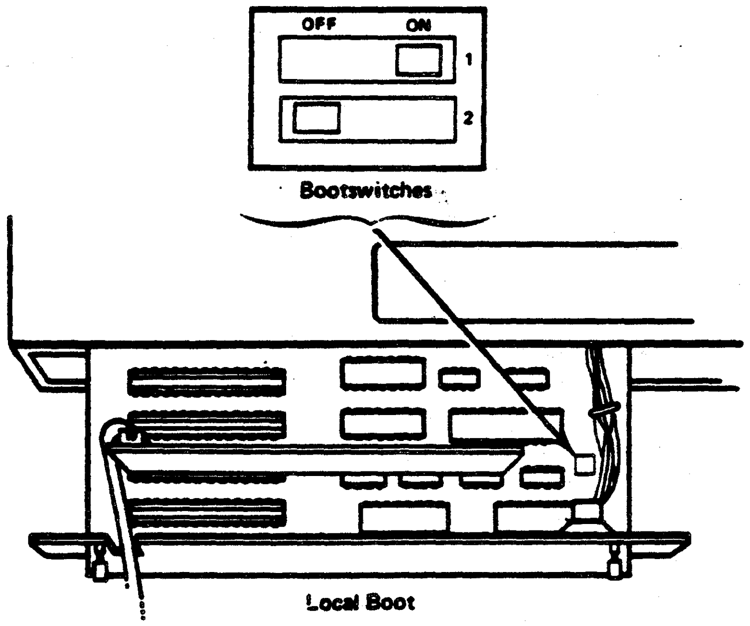
NOTE ON LOADER ERRORS

If you receive a message "Loader error 1" while trying to execute a program, your concept does not have enough memory to execute the program. Loader errors other than 1 indicate the file that contains the program is faulty.

BOOT SWITCHES

If the boot switches described in The Corvus Concept Personal Workstation User Guide are different than the ones you find in the pull-out drawer, your Corvus Concept has an earlier version of the processor board.

In this case, your boot switches are located on the right side of the drawer. There are only two switches which look like the following:



SWITCH 1	SWITCH 2	TYPE OF BOOT
Off	Off	Boot from diskette
On	On	Prompt for type of boot
Off	On	Boot from Omninet
On	Off	Boot from local hard disk

GENERAL NOTES

The following is a summary of changes for the Concept software release 18-Feb-83.

1. All programs using unit CCwndIO must be recompiled and relinked with the new CCLIB library.

2. All programs using UnitRead, UnitWrite, or UnitStatus must be recompiled.

3. Programs compiled under a version of the Concept Operating System with a date of February 1983 and later are not compatible with previous versions of the Operating Systems. Programs compiled on the later versions of the Operating System will not run on earlier versions of the Operating System. In some cases this will suspend the entire system which can be remedied only by powering off the Workstation and powering it on again.

4. Not all parts of the FORTRAN library have been tested. The routines in the CCcrt unit do not work at this time. Also, we would appreciate any input about problems with the other units in this library. (See Program Samples on second page following.)

5. LogiCalc files must be converted using the LC.CONVERT program.

6. Old character sets must be converted using the CS.CONVERT program.

7. On a 256k system in vertical orientation with floppy, printer and datacomm drivers loaded and with the 7 x 11 alternate character set, 78k code space and 48k data space is available.

8. Rules for ESC key processing:
   - Two escape characters are returned when using UnitRead or reading SYSTERM (unit 2).
   - One escape character is returned when reading INPUT or reading CONSOLE (unit 1).

   As a side note: the cursor keys return two characters when pressed; the first is an escape character and the second is A, B. C, or D.

Changes to CCLIB.OBJ functions and procedures since the last release include:

pOScurKbd	- get current keyboard record pointer
pOSsysVol	- get system volume name string pointer
pOScurVol	- get current volume name string pointer
pOSsysVrs	- get OS version number string pointer
pOSsysDat	- get OS version date string pointer
OSstrmDv	- get SYSTERM unit number
OSprtrDv	- get PRINTER unit number
OSdevType	- get device type for given unit number
OSsysSize	- get system size
OScurSP	- get current system SP
OSvrtCrt	- returns TRUE if vertical orientation
KeyPress	- returns TRUE if any key is pressed
BrkPress	- returns TRUE if BREAK key is pressed

CCcrtIO	- CrtAction GrafMode renamed to GrfMode
	- CrtAction TextMode renamed to TxtMode
CCdcpIO	- new unit (replaces CCprtIO)
CClblIO	- CClblIOterm function added (for p-System compatibility)
CComnIO	- UCdta field added to result record
CCprtIO	- obsolete
CCwndIO	- window records are 48 bytes long

SAMPLE PROGRAMS

Some people have been having problems with the gotoxy call to the crt driver. The X and Y parameters must be bytes, i.e. binary 8 bit numbers. If you have a y equal to 30 and use an integer field in the format statement, the FORTRAN formatter will convert this to an ASCII 3, followed by an ASCII 0. The following are some examples of using commands.

FORTRAN

```
      PROGRAM FTEST
C THIS WILL PERFORM A GOTOXY
      INTEGER*1 X,Y,ESC,EQUL
      X=20
      Y=15
      ESC=27
      EQUL=61
      WRITE(*,900)ESC,EQUL,X,Y
 900  FORMAT(4A1\)
      WRITE(*,1000)
 1000 FORMAT('HEY LOOK WHERE IM AT'\)
      END
```

Pascal

```
PROGRAM TEST;

VAR X,Y,MODE:INTEGER;

PROCEDURE ORG(X,Y,MODE:INTEGER); {PROCEDURE TO SET THE ORIGIN}
BEGIN
  WRITE(CHR(27),'o');
  UNITWRITE(36,X,2);
  UNITWRITE(36,Y,2);
  WRITE(CHR(MODE));
END;

PROCEDURE GOTOXY(X,Y:INTEGER);
BEGIN
  WRITE(CHR(27),'=',CHR(X),CHR(Y));
END;

PROCEDURE DRAWLINE(X1.Y1.X2.Y2.MODE:INTEGER);
BEGIN
   WRITE(CHR(27),'l');
   UNITWRITE(SCRNUNIT,X1,2);
   UNITWRITE(SCRNUNIT,Y1,2);
   UNITWRITE(SCRNUNIT,X2,2);
   UNITWRITE(SCRNUNIT,Y2,2);
   WRITE(CHR(MODE));
END;

BEGIN
END.
```

VOLTAGE SELECTION FOR THE CONCEPT MONITOR

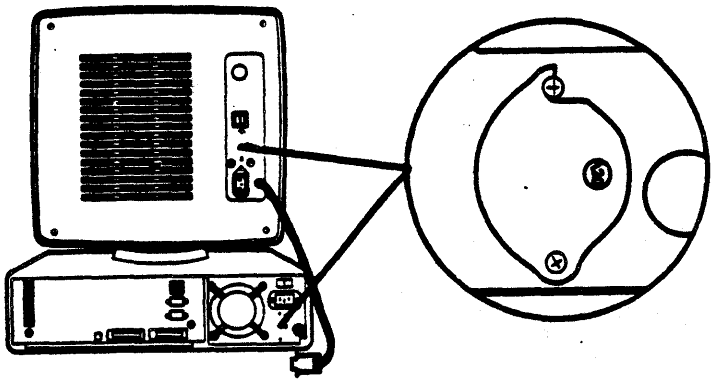
Two different voltage selection switches have been shipped with Concept monitor and base units. One is a six position switch that has the following selections:

150   220   130   260   110   240

The other is a four position switch that has the following selections:

220   120   240   110

NOTE: The four position switch has been mislabeled. The 240 VAC position is not wired to the power supply.



Voltage Settings

To correct the 240 VAC switch selection proceed as follows:

1. Turn off the power switches on the base and monitor.
2. For 240 VAC selection, place the selection arrow at 110 VAC, then turn the selection switch one position in a counter-clockwise rotation. This is an unmarked position, but it is wired to the power supply rated for 240 VAC.

UPGRADING THE CORVUS CONCEPT OPERATING SYSTEM TO VERSION 1.1a

The latest version of the Corvus Concept Operating System (CCOS) can be easily installed to work with your present system if you have version 1.0C of CCOS. If you do not have version 1.0C software on your system, you will either have to sysgen your disk using the new release diskettes or ask Corvus to send you a version 1.0C upgrade package.

Updating the Operating System

The first step in updating the operating system is to make sure you have read-write access to both the CCSYS and CCUTIL volumes. Make sure there are no personal files or non-Corvus-supplied files in the CCSYS or CCUTIL volumes. If there are any there may not be enough space to store all of the system files. Transfer any non-system files to another volume before going any further with these steps.

a) Place the diskette named FBOOT in the diskette drive. Hold down the [**COMMAND**] key and press [**ExecFile**]. In response to the prompt type the filename **/FBOOT/PURGE.1.0C** and press [**RETURN**]. Be sure to type the name accurately; the name ends with <u>zero C</u>, not <u>Oh C</u>.

   The file PURGE.1.0C deletes files from the operating system version number 1.0C thereby quickening the update procedure.

b) With the FBOOT diskette still in the diskette drive hold down the [**COMMAND**] key and press [**ExecFile**]. In response to the prompt type the filename **/FBOOT/SYSTEM.UPDATE** and press [**RETURN**]. The following displays in the system window:

```
--------------------------------------------------------

  File ID: /FBOOT/SYSTEM.UPDATE

  SYSTEM.UPDATE assumes volumes /CCSYS and /CCUTIL exist
  and have read/write access

Continue? [Y/N]:

--------------------------------------------------------
```

Press **Y** for yes, and answer any other questions with **Y** for yes. The files that are being transferred to CCSYS already exist, so the update asks if it should delete the old version of each file. The Exec file called PURGE.1.0C that was executed in the first step deleted the files that are not needed to boot. The other diskettes do not have as many duplicate files, and therefore copy faster than the FBOOT diskette.

c) Reboot the Concept after updating with the FBOOT diskette, then continue upgrading the software by repeating step b) above for each diskette received substituting the diskette's name for FBOOT when typing the Exec file name. The upgrade should proceed in the following order: FBOOT, FSYSGEN, FCCSYS1, FCCSYS2, FCCSYS3, FCCSYS4, FPSYS, FEDWORD, followed by the optional software packages FLGICLC, FPASCAL, and FFORTRN.

After all new diskettes have been processed reboot the Concept, and make sure the updated software is on the disk. Then continue with the next step.

Updating the Boot File

The boot file is in the CORVUS volume which is not normally mounted, and you may not have have read-write access. **This step can be dangerous to your files if you do not follow the directions carefully.** The first step is to be sure you have read-write access to the CORVUS volume. Because of how the Boot Manager is written, if you do not have read-write access you can remove the old file BOOT.CONCEPT but you cannot add the new one. If that happens it is not possible to boot from the drive.

a) Enter the Access manager to mount the CORVUS volume with read-write access for SMGR.

Press **[Const II]**
Press **[Drive Manager]**
Press **[Access Manager]**

Type in the appropriate passwords to enter the Access Manager.

Type **SMGR**
Press **[RETURN]**

Press **[Change Access]**

Type **CORVUS**
Press **[RETURN]**

Type **RW**
Press **[RETURN]**

Type **MOUNTED**
Press **[RETURN]**

Press **[RETURN]**

Press **[RETURN]**

Press **[SPACE]**

Press **[Exit]**

Press **[Exit]**

b) Re-boot the system logging on as SMGR. Then press ! and press **[ListVol]**. The exclamatin point ListVol will list all the volumes mounted and their read-write status. Make sure that the volume CORVUS is listed there with RW to the right. If not return to the Access Manager and mount the CORVUS volume with read-write access. Then be sure the file BOOT.CONCEPT transfered to CCUTIL properly during the upgrade. To do this press **[ListVol]**, type **/CCUTIL** and press **[RETURN]**. The file BOOT.CONCEPT should be in the volume.

c) Follow the sequence of these steps exactly:

Press **[Const II]**
Press **[Drive Manager]**
Press **[Boot Manager]**

At this point the Constellation II program checks for the slot number if there is more than one drive available. If the prompt has the wrong slot as the default, enter the slot number and continue.

Press **[RETURN]**

Now the Constellation II program checks for the proper passwords to enter the Boot Manager routine. Type the appropriate passwords and press **[RETURN]**.

Press **[Remove File]**

At this point the program asks for the type of boot file to be removed.

Type **CONCEPT**
Press **[RETURN]**

The program asks the following question:

```
----------------------------------------------------------

OK to remove boot file /CORVUS/BOOT.Concept (Y/N) Y

----------------------------------------------------------
```

Press **[RETURN]**

A message displays saying that the file has been purged, and that you are to press <space> to continue.

Press **[SPACE]**
Press **[Add File]**

A prompt appears asking for the name of the boot file.

Type **/CCUTIL/BOOT.CONCEPT**
Press **[RETURN]**

The program asks for the type of computer being used.

Type **CONCEPT**
Press **[RETURN]**

The program asks the following question:

```
----------------------------------------------------------

OK to add boot file /CORVUS/BOOT.Concept (Y/N) Y

----------------------------------------------------------
```

Press **[RETURN]**

A message displays saying that the file has been added, and that you are to press <space> to continue.

Press **[SPACE]**
Press **[Exit]**
Press **[Exit]**

The Dispatcher level should reappear, and the boot file has been added.

HOW TO CONVERT OLD .LCAR FILES TO NEW .LC FILES FOR USE WITH LOGICALC

Versions of Logicalc numbered 1.1 and earlier created data files with a .LCAR suffix. LogiCalc versions 2.0 and later create data files with a .LC suffix. Files with a .LCAR suffix must be converted using the LC.CONVERT program before they can be used with the versions numbered 2.0 or later. The conversion process is as follows:

1. Locate the old .LCAR file, and the LC.CONVERT program. LC.CONVERT should be in the volume /CCUTIL, or on the distribution diskette FLGICLC dated February 18, 1983.

2. Run the LC.CONVERT program from the Dispatcher level of the Concept. To do this

Type	/CCUTIL/LC.CONVERT
Press	[RETURN]

3. The following prompt appears in the System window:

   ```
   -------------------------------------------------------

   LogiCalc File Conversion Program 1.1
   Input file name:_

   -------------------------------------------------------
   ```

4. Enter the name of the .LCAR file to be converted, but do not type the suffix. Use the following format for the file named OLDFILE.LCAR in the volume /VOL1:

Type	/VOL1/OLDFILE
Press	[RETURN]

5. The prompt for the new file appears:

```
--------------------------------------------------------

Output file: _

--------------------------------------------------------
```

6. Enter the new file name for the character set. It is a good idea to keep the same name and let the system add the CSD prefix. This way the name will still be easy to remember.

```
        Type    /VOL1/OLDFILE
        Press   [RETURN]
```

7. When the conversion is finished, the function key labels reappear at the bottom of the screen, and the select function prompt appears in the Command window. The new file can be used by the display driver, and the character set editor. The old files are now obsolete.

HOW TO CONVERT OLD CSH AND CSV FILES TO NEW CSD FILES FOR USE WITH THE DISPLAY DRIVER

Versions of the Corvus Concept Operating System (CCOS) numbered 1.1a or later do not recognize the old CSH and CSV character sets. The new version of the display driver accepts one character set with a new format and a new prefix--CSD. Old character sets must be converted to the CSD format before the Concept will load them. If you have both a CSH and a CSV version of a character set, only one of the character sets needs to be converted, since the new driver uses only one type. To convert the character set, perform the following operations:

1. Locate either the CSH or CSV file, and the CS.CONVERT program. CS.CONVERT should be in the volume /CCUTIL, or on the distribution diskette FCCSYS3 dated February 18, 1983.

2. Run the CS.CONVERT program from the Dispatcher level of the Concept. To do this

Type	/CCUTIL/CS.CONVERT
Press	[RETURN]

3. The following prompt appears in the System window:

    ```
    ------------------------------------------------------------
    Input file:_
    ------------------------------------------------------------
    ```

4. Enter the name of the old CSH or CSV file to be converted, but do not type the suffix. Use the following format for a file named CSH.OLDFILE or CSV.OLDFILE in the volume /VOL1:

Type	/VOL1/OLDFILE
Press	[RETURN]

5. The prompt for the new file appears:

```
-----------------------------------------------------

Output file: /VOL1/OLDFILE

-----------------------------------------------------
```

6. The same file name can be used, and a new file with a .LC suffix is created. A new file name can also be used, in the form /VOL2/NEWNAME.

   Press [RETURN]

7. When the conversion is finished, the function key labels reappear at the bottom of the screen, and the select function prompt appears in the Command window. The new file has a .LC suffix and can be loaded by the LogiCalc program.

SVS PASCAL AND FORTRAN ENHANCEMENTS

The following new features and enhancement have been added to SVS Pascal in version 1.1. Unless otherwise noted, all changes are compatible with version 1.0.

1. **NEW** and **DISPOSE** now take a 4 byte length as a parameter specifying the amount of memory to be allocated or returned. The new library entry point names are %_NEW4 and %_DISP4. The old names %_NEW and %_DISP are still in the library and take 2 byte length parameters.

2. An improved 4-byte integer divide/mod routine has been included. It is about 3 times faster than the old one.

3. The Boolean operators **AND** , **OR** and **NOT** can now be applied to **INTEGER** values. The meaning is to **AND**, **OR** and **NOT** the binary integer values contained in the operands. The type of the result is **INTEGER**.

4. Pascal now accepts arrays larger than 32K bytes.

5. SVS Pascal now accepts more than 32K bytes of variables in all scopes (except that there can be at most 30K bytes of **VALUE** parameters to any procedure or function).

6. Double precision has been added to SVS Pascal. The name of the data type is **DOUBLE**, and double precision values are denoted by a **D** as an exponent instead of the normal single precision **E**, as in 3.14159D0. All standard trig routines are generic, that is if the type of the argument is **REAL** then the single precision routine is called, and if the argument is **DOUBLE** then the double precision routine is called. Likewise **TRUNC**, **ROUND** and automatic conversion from **INTEGER** to **DOUBLE** is done. To convert from **DOUBLE** to **INTEGER** either **TRUNC** or **ROUND** must be called explicitly, as is the case for **REAL**. Arithmetic is carried out at the higher mode of the operands. That is **A + B** is done in **DOUBLE** if either operand is **DOUBLE**, otherwise it is done in **REAL** if either is **REAL**, otherwise **LONGINT** or **INTEGER**. All other **REAL** features are available to things of type **DOUBLE**, such as reading and writing via **TEXT** files, etc.

7. Variable names are now considered significant in the first 31 charaters. This applies only to compile time names. Link time names are still only significant in the first 8 characters.

8. In addition, **UNITREAD** and **UNITWRITE** now take 4 byte parameters for both the block number and byte counts. **UNITSTATUS** also requires a 4 byte value parameter instead of the old 2 byte one. This change is **NOT**

backwards compatible. Consequently all new programs using these routines cannot run on an older version of the system.

The following new features and enhancement have been added to SVS FORTRAN in version 1.1. Unless otherwise noted, all changes are compatible with version 1.0.

1. If the **$CHAREQU** compiler directive has been selected, then it is possible to initialize non-character variables with **CHARACTER** values in **DATA** statements. The initialization is carried out as if the variable was of type **CHARACTER*n**, where n = the number of bytes occupied by the variable. Consequently, the statements:

   ```
   INTEGER I,J
   DATA I,J/'ABCDEFGH'/
   ```

   does not initialize I to **'ABCD'** and causes a compile time error stating that there is not enough data. This is consistent with the way that **CHARACTER** variables are initialized. Also note that:

   ```
   INTEGER I
   DATA I/'A'/
   ```

   initialized I to the value 'A   '.

2. Formatted **READ** and **WRITE** now allows all data types to be read and written using an **A** of **An** format descriptor. An n byte variable is considered to be type **CHARACTER*n** in this instance.

3. A new intrinsic function **PEEK** has been added. It is of type **INTEGER*1** and takes a parameter of type **INTEGER*4**. It returns the value of the byte at the memory location specified by its parameter. Please note that **INTEGER*1** represents signed values.

4. A new library routine **POKE(IVAL,IADDR)** has been added. It stores the **INTEGER*1** value **IVAL** into the memory location specified by the **INTEGER*4** value **IVAL**.

5. A new library routine **VERS** with no parameters can be called. It prints the **FORTRAN** library date and version number on the standard output device.

6. Compiler command line flags +b, +c72 and +x have been added. They behave exactly as if the compiler directives **$BIGCODE**, **$COL72** and **$XREF** have been specified.

MIXING CONSTELLATION II AND CONSTELLATION I SYSTEMS

The following procedure explains how to add Constellation II software for the Corvus Concept to an existing Constellation I Apple system. You should be thoroughly familiar with the Constellation I system.

Before beginning this procedure, we recommend that you backup the information on your drive.

1. Log-on to your Apple as SMGR.

2. Execute the program **VMGR** and list the volumes on your drive. For all the volumes on drive 1, write down the volume names, their starting block addresses, and their lengths.

3. You will need room on drive 1 for three new system volumes that will be created by Constellation II. This will be 4296 blocks of contiguous space.

   If you do not have this much free space already, you must reallocate some of the space you have previously allocated for Apple volumes.

4. Exit the **VMGR** program, and backup all volumes that fall within the area where the Constellation II volumes will be placed.

5. When you have backed up this area, execute the **VMGR** program and delete the volumes you just backed up. Create a volume called CONSTII that is 4296 blocks in size. Write down the starting address of this volume.

   Exit the **VMGR** program.

6. Follow the instructions in "The Corvus Concept Disk Drive Installation Guide" to initialize your drive with Constellation II software.

   When you are prompted for the starting block address of the CORVUS volume, enter the starting block address of the CONSTII volume that you wrote down in step 5. When you are prompted for the length of this volume, enter 200.

When prompted for the starting block address of the CCSYS volume, enter the sum of CONSTII + 200. For the length, enter 2048.

When prompted for the starting block address of the CCUTIL volume, enter the sum of CONSTII + 200 + 2048. For the length, enter 2048.

Finish the Constellation II initialization procedure.

7. For Constellation II to recognize existing Apple volumes on the disk, you must create Constellation II volumes over the Apple volumes (for Apple volumes on the second half of a 20 MB drive, see VDO TABLES at the end of this section). You must use the volume names, addresses, and starting block addresses that you wrote down in step 2.

   While creating these volumes with Constellation II, make them UCSD/PASCAL type and do not format them when you are prompted. Not formatting these volumes leaves the information in them intact and usable from Apples and Constellation II.

   The three Constellation II system volumes CORVUS, CCSYS, and CCUTIL cannot be shared.

   Follow the instructions in "The Corvus Concept System Manager's Guide" to create volumes, users, and user accounts. When creating new Constellation II volumes, you should first create them on your Constellation I system and then use the same address to create the volume for Constellation II. Do not format the volume if it contains data.

   The Constellation I and Constellation II network software does not know about each other's tables. The two systems can co-exist only because as system manager you keep both sets of tables the same. Some very important points to keep in mind are the following:

   a) The first 1024 blocks (8 - 1031) are used for the Constellation I system volume and associated tables.

   b) Block 8 is used by Constellation II to point at the CORVUS volume. The CORVUS volume contains all Constellation II tables.

   c) Always create volumes on Constellation I first since a volume created on Constellation I destroys all data.

d) When creating a Constellation II volume over a Constellation I volume, never format the volume. If you do, it will destroy any data in the volume.

It is recommended that you add Constellation II to an existing Constellation I Apple system; however, if necessary, the following procedure explains how to add Constellation I software to an **existing** Constellation II Concept system. You should be thoroughly familiar with the Constellation II system.

Before beginning this procedure, we recommend that you backup the information on your drive.

You must have Constellation I software with a release date of 21-OCT-82 or later.

1. Log-on to your Concept as SMGR

2. Use the Volume Manager function of Const II to list the volumes on the drive. If you have a volume starting at block 9, you must remove all files from this volume then remove this volume.

   Constellation I uses the first 1024 blocks for a volume named SYS. Create a volume that starts at block 9 and is 1023 blocks long and call it APSYS.

3. When initializing for Constellation I, you must be careful not to destroy the Constellation II information on your disk.

   Follow the instructions for initializing the drive for Constellation I found in the "Omninet Disk System Installation Guide." When you run the **PSYSGEN** program, it begins by asking the following:

```
+-----------------------------------------------------+
|                                                     |
|  OK TO DESTROY ALL DATA                             |
|  ON DISK IN ALTERNATE SLOT [Y/N]?                   |
|                                                     |
+-----------------------------------------------------+
```

   In response to this prompt, you must

   Press **Y**

4. Continue with the initialization procedure. When you are finished initializing for Constellation I, you may create volumes with Constellation I; however, you must be careful not to create a Constellation I volume over an existing Constellation II volume. If you do, it will destroy all data in the Constellation II volume.

5. Write down all the volume names, lengths, and starting block addresses of the Constellation I volumes you create.

6. For Constellation II to recognize the Constellation I volumes on the disk, you must create Constellation II volumes over the Constellation I volumes. You must use the volume names, lengths, and starting block addresses that you wrote down in step 5.

   While creating these volumes with Constellation II, make them UCSD/PASCAL type and do not format them when you are prompted. Not formatting these volumes leaves the information in them intact and usable from both Constellation I and Constellation II.

   The three Constellation II system volumes CORVUS, CCSYS, and CCUTIL cannot be shared and you must **NOT** create a Constellation I volume that would overlap any of these volumes.

VDO TABLES

When creating Constellation II volumes over Constellation I volumes that are on the second half of a 20 MB drive, you will the have to convert the addresses the VMGR program gives you to addresses that Const II understands.

The following table shows the block number equivalences for Constellation I and Constellation II. In the first entry below, the VMGR program lists a block address on virtual drive 2 as being 0, this address for Constellation II is 18222. For the first entry below, to figure where block 8 on virtual drive number 2 would be for the concept, add 8 to 18220.

REV H DRIVES		
VDO	block # for Apple	block # for Concept
911	0	18220
896	0	17920
REV B DRIVES		
976	0	19520
947	0	18940

INSTALLING AN ADD-ON DRIVE TO AN EXISTING CORVUS CONCEPT DISK SYSTEM

This upgrade note steps through the intallation of the hardware and software necessary to add a drive to an existing Corvus Concept Disk System. It is meant for users experienced with Corvus equipment and software, and the steps are not fully explained at points where a prior knowledge is assumed.

Add-on Drive Setup

Before setting up an add-on drive you must already have a functioning system. Use the Diagnostic utility to check the controller firmware version of your existing system. If the disk has a version below CF17.3, you must update your firmware. Contact the Corvus Customer Service Department or an Authorized Corvus Service Center for a free update.

a) Power off all equipment.

b) Find out the add-on drive's unit number. The size of the preceding drives in the daisy chain determine the drive's unit number. Each six Mbyte and ten Mbyte drive counts as one drive. A twenty Mbyte, although one physical drive, counts as two drives.

For example:

If you have a twenty Mbyte drive and you are adding a ten Mbyte drive to your system, the ten Mbyte drive becomes the third unit in the system. Its unit number is three.

If you have a ten Mbyte drive and you are adding a twenty Mbyte drive to your system, the twenty Mbyte drive becomes the second unit in the system. Its unit number is two.

---

c) When you have figured out the unit number for the add-on drive, set the dip switches on the drive's controller board.

Remove the screws that hold down the cover of the drive cabinet. Lift the cover and turn it to expose the controller board, which is attached to the underside of the cabinet cover.

Use the end of a bent paper clip, or something of about the same size, to set the switches. The following charts give the appropriate switch settings for each unit number. An X shows which side of the switch you depress.

<u>**Switch settings for REV B and REV C drives**</u>

```
 UNIT                                UNIT
NUMBER          SETTING             NUMBER          SETTING

              +--------+                          +--------+
              |12345678|                          |12345678|
   2          |x x     |              5           | x      |
              | x xxxxx|                          |x xxxxxx|
              |  OPEN  |                          |  OPEN  |
              +--------+                          +--------+

              +--------+                          +--------+
              |12345678|                          |12345678|
   3          |x       |              6           |  x     |
              | xxxxxxx|                          |xx xxxxx|
              |  OPEN  |                          |  OPEN  |
              +--------+                          +--------+

              +--------+                          +--------+
              |12345678|                          |12345678|
   4          | xx     |              7           |        |
              |x  xxxxx|                          |xxxxxxxx|
              |  OPEN  |                          |  OPEN  |
              +--------+                          +--------+
```

<u>**Switch settings for H series drives**</u>

```
 UNIT                                UNIT
NUMBER          SETTING             NUMBER          SETTING

              +--------+                          +--------+
              |12345678|                          |12345678|
   1          |x  x    |              3           |   x    |
              | xx xxxx|                          |xxx xxxx|
              |  OPEN  |                          |  OPEN  |
              +--------+                          +--------+

              +--------+                          +--------+
              |12345678|                          |12345678|
   2          |x       |              4           |        |
              | xxxxxxx|                          |xxxxxxxx|
              |  OPEN  |                          |  OPEN  |
              +--------+                          +--------+
```

d) When the switches are set, close the drive cabinet.

e) Connect the add-on drive using the special add-on flat cable. Connect one end of this cable to the PROCESSOR port on the back of the add-on drive cabinet. Connect the other end of this cable to the port, labeled DRIVE.

on the back of the preceding drive.

Cables always exit downward from the back of the drive cabinet with the stripe to your right as you face the back of the drive.

f) Set the front panel switches for the add-on drive to the same settings as the first drive. For example, if the drives are connected to a multiplexer network, the multiplexer switch must be on for all drives.

Initializing an Add-on Drive

The procedure for initializing an add-on drive is very similar to initializing the first drive on the chain. The difference is that the add-on drives do not need to have the operating system installed.

a) **Type /CCUTIL** and press **[SetVol]**.

b) When the volume is set to CCUTIL, type **SYSGEN**. To invoke the sysgen program you must also type the password **HAI**.

c) Press **[Init Drive]**. The sysgen program asks for the drive number. Press **2** and press **[RETURN]**. The next questions ask for the drive name and password which can be any valid name you wish.

d) The next step asks for information about the volume CORVUS. The default on both questions should be acceptable.

e) After answering those questions you are prompted for whether or not you would like to install the operating system. You do not want to install it on any add-on drives since it is already on the first drive. Press **N** for no and press **[RETURN]**.

f) The program then asks if it is OK to continue. The default is no, so you must press **Y** for yes and press **[RETURN]**.

g) After a minute or so the Command window displays a message telling you to press [SPACE] to continue. Press **[SPACE]**. and press **[Exit]**.

Creating Volumes on an Add-on Drive

To create a volume on an add-on drive is essentially the same as adding volumes to the first drive. The difference is that when entering Const II, the add-on drive number, name, and password must be specified. Users, and user access tables are always created and maintained on drive 1.

Corvus Concept

DISPLAY DRIVER

ERS

(External Reference Specification)

TABLE OF CONTENTS

1. Introduction

This document describes the features provided by the Corvus Concept screen driver. These features are made available to be used by many different software packages, but this document only describes the basic feature set provided. Calls to the CRT driver are made through the UNITxxxxx calls, for example, UNITWRITE. Different languages may have different calling protocols (e.g. WRITELN for Pascal), so you may need to consult other manuals if you are not working directly with the screen driver. In general, though, the information in this document may be used quite readily. For example, the following Pascal statement will cause the bell tone to sound:

```
    write( chr(7) );
```

The following Pascal statement will cause the current window to clear:

```
    write( chr(27), 'J' );
```

2. UNITWRITE

The unitwrite command is used to send bytes to the display. Normally, all bytes sent will be displayed as the corresponding ASCII character. For example, sending the decimal byte value of 65 to the screen via unitwrite will cause the letter 'A' to appear on the screen at the current cursor location. There is a group of bytes, however, that is interpreted specially. These special codes are listed in the following table:

Command Sequence	Hex Codes	Description
Ctl-G	07	bell
Ctl-H	08	cursor left (backspace)
Ctl-I	09	tab (8 spaces)
Ctl-J	0A	cursor down (linefeed)
Ctl-K	0B	cursor up
Ctl-L	0C	cursor right
Ctl-M	0D	carriage return
ESC = col row	1B,3D,col,row	gotoxy
ESC A	1B,41	cursor up
ESC B	1B,42	cursor down (linefeed)
ESC C	1B,43	cursor right
ESC D	1B,44	cursor left (backspace)
ESC E	1B,45	insert line
ESC G 0	1B,47,30	set video to normal
ESC G 4	1B,47,34	set video to inverse
ESC G 8	1B,47,38	set video to underline
ESC G <	1B,47,3C	set video to inverse+underline
ESC H	1B,48	cursor home
ESC J	1B,49	clear window + home cursor
ESC K	1B,4B	clear to end of line
ESC O c1 c2	1B,4F,c1,c2	overstrike c1, c2
ESC Q	1B,51	insert character
ESC R	1B,52	delete line
ESC W	1B,57	delete character
ESC Y	1B,59	clear to end of window
ESC Z	1B,60,N	invert N characters
ESC a	1B,61	page mode on
ESC b	1B,62	turn off cursor
ESC c	1B,63	turn on cursor
ESC d	1B,64	enter key: carriage return
ESC e	1B,65	character enhancements
ESC f	1B,66	*fill block
ESC g	1B,67	graphics mode
ESC i	1B,69	back tab (8 spaces)
ESC l	1B,6C	*draw line
ESC m	1B,6D	*copy block
ESC n	1B,6E	turn off scrolling
ESC o	1B,6F	*set graphics origin
ESC p	1B,70	*plot point
ESC q	1B,71	insert mode
ESC r	1B,72	insert mode off
ESC s	1B,73	turn on scrolling
ESC t	1B,74	text mode
ESC u	1B,75	underscore cursor
ESC v	1B,76	inverse cursor
ESC w	1B,77	wrap at end & beg. of line
ESC x	1B,78	no wrap at end & beg. of line
ESC y	1B,79	page mode off
ESC z	1B,7A	invert screen

*Graphics functions:

Function	Parameters	Byte count
Set origin	x,y,qualifier	7 (11221)
Plot point	x,y,mode	7 (11221)
Draw line	x1,y1,x2,y2,mode	11 (1122221)
Fill block	x,y,height,width,density	11 (1122221)
Copy block	x1,y1,height,width,x2,y2	14 (11222222)
WriteBytes	(see UnitStatus)	
ReadBytes	(see UnitStatus)	

mode	<0 invert, =0 clear, >0 set
density	1 dense, 2 less dense, etc.
qualifier	1 rel to graphics origin 2 abs graphics origin 3 rel to cursor position 4 abs text origin

2.1. Overstrike

The overstrike enhancement is specified as follows:

```
ESC O <char1> <char2>
```

<Char1> and <char2> are OR'ed together at the cursor position. If a character is not in the character set, a blank is substituted. Only two characters may be overstruck at one cursor position.

2.2. Character Enhancements

Old style enhancements were supported through the "ESC G" control sequence. This control sequence is still supported, but new software should use the "ESC e" sequence. The "ESC e" sequence provides many more enhancements and will eventually be adopted as the standard:

```
ESC e <byte>
```

<Byte> is a bit pattern of 7 flags, where 1 means the feature is on, 0 - off.

x1xxxxxx

bit	flag
0	bold
1	strike out
2	inverse
3	underline
4	superscript
5	subscript
6	(always on)
7	double underline

Note that inverse and underline are already implemented in the current CRT driver, using the sequence ESC G char. This sequence will continue to be supported, for compatibility reasons.

Two pairs of enhancements are mutually exclusive: superscript and subscript, underline and double underline. If both flags of a pair are set, the flag with the lower bit number takes precedence. The order of checking flags and applying enhancements to the character is:

1) super/subscript
2) bold
3) strikeout
4) double/underline
5) inverse

The algorithms used for displaying the enhancements are as follows:

BOLD - character is OR'ed over itself one dot position to the right in the character cell.

SUPERSCRIPT - character is shifted up two dots and the top three rows of the character cell are ORed together to make room for the superscripted character.

SUBSCRIPT - character is shifted down two dots and the bottom three rows of the character cell are ORed together to make room for the subsripted character.

UNDERLINE - the bottom row of the character cell is filled with dots.

DOUBLE UNDERLINE - the bottom two rows of the character cell are filled with dots.

STRIKEOUT - the fifth row of the character cell (first row is row one) is filled with dots.

The character set enhancements will not work in any cell smaller than 7 by 11 dots. The default character set is 7x11.

The window record has been expanded to accommodate the extra flags. User programs that create windows will have to be re-compiled so that they allocate space for the larger window record. Otherwise program data may be clobbered. This change was effective Jan. 1, 1983.

3. UNITSTATUS

Some driver functions are not accessible by using the unitwrite function. These other functions are accessed by using the UNITSTATUS call. The format of the call is:

```
UnitStatus(DisplayUnitNo,Buffer,Func);
```

where DisplayUnitNo and Func are integers and Buffer is a parameter block containing the parameters in the order shown.

Function	Code	Parameters
Read cursor position	0	xposition,yposition: integer;
Create window	1	NewWindowRec: WndRcd;
Delete window	2	WindowRec: WndRcd;
Select window	3	WindowRec: WndRcd;
Clear window	4	WindowRec: WndRcd;
Get window status	5	homex,homey,width,height: integer;
WriteBytes	6	bytecount: integer; pBuff: pBytes;
ReadBytes	7	bytecount: integer; pBuff: pBytes;
Load CRT Table	8	see below

3.1. LOADABLE CRT TABLES

The new CRT driver will provide user-loadable translation tables. This means that the CRT function control codes can be changed to emulate other terminals at the driver level. The CRT driver contains a default translation table which is in effect when the system is booted. You can switch to a different table by loading the new table into memory and passing its address to the CRT driver to make it the current table:

```
UnitStatus(CRTunitnumber,Table,8);
```

where Table is the new translation table. Since the second parameter in the UnitStatus procedure is a VAR parameter, the CRT driver receives the address of the new table. If this address is nil (0) the default table becomes current.

The ASCII value of the control character is used as an index into the table. Nonnegative bytes in the CRT table correspond to entries in the jump table of the CRT driver. Table indices from 0..$1F (first 2 rows) refer to control characters; table indices from $20..$7F refer to the character after an ESC. A byte value of $FF means the code is invalid or not implemented. This is the default table contained in the CRT driver:

```
   | 0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
===|===============================================
 00| FF FF FF FF FF FF FF 08 03 06 01 00 02 05 FF FF
 10| FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
 20| FF FF FF 26 FF FF FF FF FF FF FF FF FF FF FF FF
 30| FF FF FF FF FF FF FF FF FF FF FF FF FF 24 FF FF
 40| FF 00 01 02 03 0E FF 11 04 FF 09 0A 14 15 FF 2C
 50| 26 0C 0F FF FF FF FF 0D FF 0B 2E FF FF FF FF FF
 60| FF 16 19 18 05 2D 29 1A FF 07 FF FF 28 2A 1D 2B
 70| 27 1E 1F 1C 1B 20 21 22 23 17 10 FF FF FF FF FF
```

Each byte in the CRT table refers to a driver function. Here is the list of available functions (codes are in hex). The default sequence follows.

HEX	FUNCTION DESCRIPTION	DEFAULT SEQUENCE(S)
00	cursor up	ESC A or CTL-K
01	cursor down (linefeed)	ESC B
02	cursor right	ESC C or CTL-L
03	cursor left (backspace)	ESC D or CTl-H
04	cursor home	ESC H
05	carriage return	CTL-M
06	tab	CTL-I
07	back tab	ESC i
08	sound bell	CTL-G
09	clear screen and home cursor	ESC J
0A	clear to end of line	ESC K
0B	clear to end of screen	ESC Y
0C	insert character	ESC Q
0D	delete character	ESC W
0E	insert line	ESC E
0F	delete line	ESC R
10	invert screen	ESC z
11	video command	ESC G mode
12	inverse video	
13	normal video	
14	set auto LF on CR	ESC L
15	suppress auto LF	ESC M
16	page mode on	ESC a
17	page mode off	ESC y
18	cursor on	ESC c
19	cursor off	ESC b
1A	graphics mode	ESC g
1B	text mode	ESC t
1C	scrolling on	ESC s
1D	scrolling off	ESC n
1E	insert mode on	ESC q
1F	insert mode off	ESC r
20	underscore cursor	ESC u
21	inverse cursor	ESC v
22	wrap at eoln	ESC w
23	no wrap at eoln	ESC x
24	CCOS type gotoxy	ESC = col row
25	CP/M type gotoxy	
26	skip two characters	ESC # c1 c2 and ESC P c1 c2
27	plot point	ESC p <parms>
28	draw line	ESC l <parms>
29	fill block	ESC f <parms>
2A	copy block	ESC m <parms>
2B	set origin	ESC o <parms>
2C	overstrike 2 chars	ESC O ch1 ch2
2D	character set enhancements	ESC e flags
2E	invert characters at cursor	ESC Z count

NEW UNITSTATUS FUNCTIONS OF THE KEYBOARD DRIVER.

1) Function code : 0 Name : Send Raw

Parameter Block :

a) 1 word Type = INPUT

This procedure sets or clears the Send Raw flag in the driver. If the parameter is 0 then the flag is cleared. If it is 1 then the flag is set.

The Send Raw flag, if set, forces the Keyboard to send all characters to the user, including Control-Q and Control-S. Normally, when the flag is clear, Control-Q and Control-S are not sent out but used to control the display.

2) Function code : 1 Name : Test and Clear Break flag

Parameter Block :

a) 1 word Type = OUTPUT

This function returns the value of the Break flag then clears it. If the parameter is 0 then the flag was cleared. If it is 1 then the flag was set.

The Break flag represents the state of the Break key. If it is clear then the Break has not been pressed. If it is set then since the last call to this function the Break key has been pressed. The flag is cleared when this function is called or when the driver is initialized.

3) Function code : 2 Name : Number of Characters Available

Parameter Block :

a) 1 word Type = OUTPUT

This function returns the number of characters available in the keyboard buffer.

USING EXEC FILES AND COMMAND LINES

This Technical Note explains the use of the Corvus Concept Exec files. Exec files, sometimes referred to as "shell script files," are used to automate the user interface with the Corvus Concept. An Exec file is a text file specifing a series of programs to be executed. When the ExecFile function is requested, these programs are executed, in order, with no user intervention. An example of this is a Pascal compilation, followed by the linker, and finally running the executable program.

Exec File Record Format.

Each Exec file record is interpreted as if it were entered on the command line by the user. The general format for a command is:

```
    command (arg1 arg2 .. argn) ( < Ifile ) ( > Ofile )
```

where:

command	- is the system command or program file name to be executed. If the volume name is not specified, the current volume is searched for the specified program file. If the file is not found in the current volume, the system volume is then searched. If the volume name is specified, only that volume is searched for the program file. A program file name with a "!" prefix indicates the program file is in the system volume. These are the same rules as requesting the program from the command line.
arg..	- are program parameters to be passed to the requested program.
Ifile	- is the default input file for the called program. If Ifile is not specified, the console is used as the default input file for the called program. If a file name is specified, text in that file is used as the default input file. If "%" is specified, the current input file (the Exec file) is used as the default input file.
Ofile	- is the default output file for the called program. If Ofile is not specified, the display is used as the default output file for the called program. If a file name is specified, default output file text is placed in that file. If "%" is specified, the current output file is used as the default output file.

Special Exec File Records

"Comment" records are identified by using a ; | { or } character as the first non-blank character in the record. Comment record are written to the output file with no further processing.

"Pause" records are identified by placing PAUSE as the first non-blank characters in the record. PAUSE may be followed by an optional text message with its context indicating a Y reply to continue or a N reply to abort command file processing. When a pause record is encountered, the text message is displayed and command file processing is suspended until a Y or N reply is indicated by the user. A Y reply continues processing and a N reply terminates command file processing.

FileMgr Parameters.

The following is a list of FileMgr commands.

```
(!CC.FILMGR) CATFIL File1 (File2 ....) > OutputFile
(!CC.FILMGR) CPYFIL -VDstVol (-D) (-Q) (-S) SrcFil1 (SrcFil2 ....)
(!CC.FILMGR) CRUNCH (-Q) /Volume1 (/Volume2 ....)
(!CC.FILMGR) DELFIL (-Q) (-T) File1 (File2 ....)
(!CC.FILMGR) FLPDIR /Volume
(!CC.FILMGR) LSTFIL File1 (File2 ....)
(!CC.FILMGR) LSTVOL (-B) (-H) (-L) /Volume1 (/Volume2 ....)
(!CC.FILMGR) LSTVOL /
(!CC.FILMGR) LSTVOL !
(!CC.FILMGR) MAKFIL NewName1[length] (NewName2[length] ....)
(!CC.FILMGR) RENFIL OldName NewName
(!CC.FILMGR) RENFIL /OldVol /NewVol
(!CC.FILMGR) SETVOL /Volume
```

SystmMgr Parameters.

The following is a list of SystmMgr commands.

```
!CC.SYSMGR SETDAT (NewDate) - display (set) current date
!CC.SYSMGR SETTIM (NewTime) - display (set) current time
!CC.SYSMGR DRVVRS
```

WndowMgr Parameters.

The following is a list of SystmMgr commands.

```
!CC.WNDMGR BOXWND             - box (unbox) current window
!CC.WNDMGR CLRWND             - clear current window
!CC.WNDMGR CSDISP FileName    - load display character set
!CC.WNDMGR CSKYBD FileName    - load keyboard character set
!CC.WNDMGR DEFSCN             - clear screen and display boarder
!CC.WNDMGR DEFTTL             - update screen date
!CC.WNDMGR REVBKG             - reverse window background
!CC.WNDMGR SCRLMD             - toggle scroll mode
```

Command Line Parameters for the Set DataCom and Printer Program.

Command line parameter form:

KEYWORD=PARAMETER

1) No blanks are permitted within a parameter.

2) Parameters and keywords can be abbreviated to their shortest unique string.

KEYWORDS	PARAMETERS	COMMENT
1) UNIT	PRINTER, DC1, DC2	PRINTER can be abbreviated to 1 letter
2) BAUD	300, 600, 1200, 2400, 4800 9600, 19200	can drop all zero (0) digits
3) PARITY	DISABLED, EVEN, ODD, MARK, SPACE	only need first letter
4) HANDSHAKE	LINE/CTS/NORMAL, LINE/CTS/INVERTED, LINE/DSR/NORMAL, LINE/DSR/INVERTED, LINE/DCD/NORMAL, LINE/DCD/INVERTED, XONXOFF, ENQACK, ETXACK	Line parameters MUST have slash (/) between names. Inverted and Normal need only first letters.
5) DATACOM {ONLY FOR PRINTER}	1, 2	only need first letter
6) CHARSIZE	7, 8	Can be abbreviated to CH.
7) ALTCHARTABLE	<file name>	Must be the complete file name. Can be abbreviated to AL.
8) ACTIONTABLE	<file name>	Must be the complete file name. Can be abbreviated to AC.

9) CPI	10, 12	Can be abbreviated to CP. Only used after UNIT set to Printer.
10) LPI	6, 8	Can be abbreviated to L. Only used after UNIT set to Printer.
11) AUTOLINEFD	ON, OFF	Sets auto line feed on or off. Can be abbreviated to AU.

Defaults:

1) DataCom 1

a) BAUD	9600
b) PARITY	DISABLED
c) HANDSHAKE	XONXOFF
d) CHARSIZE	8
e) AUTOLINEFD	ON

2) DataCom 2 without the Printer
(configured for the Epson printer)

a) BAUD	4800
b) PARITY	DISABLED
c) HANDSHAKE	LINE/DSR/NORMAL
d) CHARSIZE	8
e) AUTOLINEFD	ON

3) Printer (configured for the NEC 7700 Spinwriter)

a) BAUD	1200
b) PARITY	SPACE
c) HANDSHAKE	XONXOFF
d) CHARSIZE	7
e) DATACOM	2
g) AUTOLINEFD	ON
h) CPI	10
i) LPI	6

Requirements:

1) The UNIT parameter must be specified before any other parameters.

2) More than one UNIT may be configured on a command line. It is necessary only to change the UNIT parameter.

Example of Command Lines.

```
!CC.SETDCP UNIT=DC1 BAUD=1200 PARITY=ODD HANDSHAKE=ENQACK
```

Configure DataCom 1 to 1200 baud with odd parity and use the character handshake protocol Enq/Ack.

```
!CC.SETDCP U=P D=1 B=24 P=S H=X CH=7
```

Configure the Printer to use DataCom 1 with the Xon/Xoff character handshake protocol. DataCom 1 is configured to 2400 baud, Space transmit and ignore receive parity, and a character size of 7.

```
!cc.setdcp unit=dc1 baud=1200 unit=dc2 h=xonxoff
```

Lower case is ok. Configure DataCom 1 to 1200 baud and DataCom 2 to use the Xon/Xoff character handshake protocol.

```
!cc.setdcp unit=printer altchartable=/myvol/alttable.data
```

The unit parameter must be specified as printer before either the Alternate Character translation table or the Action table are loaded. This example loads the Alternate table file named alttable.data from volume myvol. It verifies the correctness of the table before it attaches it to the driver.

Command Line Interface for the Terminal Emulator Program.

General form: TERMINAL [unit_number] [send_auto_LF] [rcv_auto_LF]

Parameters can be specified in any order. All parameters are keyword. Their general form is:

```
Keyword=Param_Value
```

No blanks are allowed within a parameter. Parameters may be in upper or lower case letters. Parameters are seperated by blanks.

1) unit_number

Keyword:	UNIT
Values:	DC1 or DC2

DC1 means DataCom 1 is the primary terminal link.

DC2 means DataCom 2 is the primary terminal link.

There is no default for this parameter. If it is not specified the program prompts the user for the correct value. If the ESCAPE key is pressed the program exits.

2) send_auto_LF

Keyword:	SENDALF
Values:	ON or OFF
	ON means Line Feed characters are sent out the primary terminal link automatically after Carriage Return characters.
	OFF means Line Feed characters are not automatically added to the character stream.
Default:	If SENDALF is not specified on the command line the state is ON.

3) rcv_auto_LF

Keyword:	RCVALF
Values:	ON or OFF
	ON means Line Feed operations are performed on the screen automatically after Carriage Return characters.
	OFF means Line Feed operations are not automatically performed after Carriage Return characters.
Default:	If RCVALF is not specified on the command line the state is ON.

4) printer_auto_LF

Keyword:	PRNTALF
Values:	ON or OFF
	ON means Line Feed characters are sent to the printer automatically after Carriage Return characters.
	OFF means Line Feed characters are not automatically sent after Carriage Return characters.
Default:	If PRNTALF is not specified on the command line the state is ON.

Function of the Terminal Emulator Program.

Generally, any characters typed at the keyboard are sent to the datacom connection. Conversely, any characters sent from the datacom are sent to the screen.

Characters sent from the datacom can be saved in a text file while they are being displayed on the screen. Just press the [FILE] function key and the program asks for the file name. If the [ESC] key is pressed the program does not use send characters to a file. If just the [RETURN] key is pressed the program uses the default file name, "ECHO.TEXT."

Characters will be sent to the file until the user exits the program or presses the [FILE] function key again. The file is closed and will be available for spooling or editing with EdWord.

If a local printer is available, the characters sent by the datacom may also be sent to the printer. Press the [PRINTER] function key to send characters to the printer. Press it again to stop. Characters will then appear on the screen and the printer.

Characters may be sent to the printer and a file at the same time.

To exit the program press the [EXIT] function key, F10.

TWO EXAMPLES OF STARTUP FILES

EXAMPLE 1

```
; Now setting up printer parameters for the Diablo 630...
UNIT=PRINTER BAUD=2400 ACTIONTABLE=/CCUTIL/PAT.DIABLO630
;
; printer parameters set.
;
; Loading character set for enhancements...
!CC.WNDMGR CSDISP /CCUTIL/CSD.07.11
;
; enhanced character set loaded.
;
; Entering the EdWord Workspace in the volume /ONE...
!ED /ONE/W
```

NOTE: EdWord cannot be called from a startup file on 256K machines.

EXAMPLE 2

```
; Now setting up printer parameters for the C-ITOH...
UNIT=PRINTER BAUD=4800 PARITY=EVEN CHARSIZE=8 ACTIONTABLE=/CCUTIL/PAT.CITOH
;
; printer parameters set.
;
; Loading character set for enhancements...
!CC.WNDMGR CSDISP /CCUTIL/CSD.07.11
;
; enhanced character set loaded.
;
; Entering the EdWord Workspace in the volume /ONE...
!ED /ONE/W
```

NOTE: EdWord cannot be called from a startup file on 256K machines.

DATA COMMUNICATION PARAMETERS FOR PRINTERS COMMONLY USED WITH THE CONCEPT

C-ITOH

BAUD	4800
PARITY	EVEN
PROTOCOL	XON/OFF
CHARSIZE	8

DIABLO 630

BAUD	4800
PARITY	SPACE
PROTOCOL	XON/XOFF
CHARSIZE	7

EPSON MX80 & MX100

BAUD	4800
PARITY	DISABLED
PROTOCOL	LINE/DSR/NORMAL
CHARSIZE	8

__IDS PAPER TIGER__

BAUD	- 9600
PARITY	- DISABLED
PROTOCOL	- XON/XOFF
CHARSIZE	- 8

__NEC 7700 SERIES__

BAUD	- 1200
PARITY	- DIABLED
PROTOCOL	- XON/XOFF
CHARSIZE	- 8

USING THE LINKER AND LIBRARY UTILITIES

The Linker

To run the Linker, you must be at the dispatcher level.

```
    Type   LINKER
    Press  [RETURN]
```

The following example shows how to link a program called MYPROGRAM.

```
    Linker - MC68000 Linker Utility    01-Dec-82
    (c) Copyright 1982 Silicon Valley Software, Inc.

    Listing file - [RETURN]
    Output file [.obj] - MYPROGRAM
    Input file [.obj] - MYPROGRAM
    Input file [.obj] - SPEC.UNIT
    Input file [.obj] - /CCUTIL/CCLIB
    Input file [.obj] - !PASLIB
    Input file [.obj] - [RETURN]
```

The file **!PASLIB** is always the last input file specified. For more information on linking the Corvus Concept system libraries, see The Corvus Concept System Library User Guide.

To use an EXEC file to link a program, you must create a text file that has a call to the Linker followed by file names. The following example shows how you could link MYPROGRAM using an EXEC file.

First, create and save a text file with the following information in the following order:

```
    LINKER < %

    MYPROGRAM
    MYPROGRAM
    SPEC.UNIT
    /CCUTIL/CCLIB
    !PASLIB
```

If we save this file with the name LNKMINE.TEXT, we may use it in the following manner:

```
    Press   [ExecFile]
    Type    LNKMINE
    Press   [RETURN]
```

The Library

To run the Library, you must be at the dispatcher level.

```
Type   LIBRARY
Press  [RETURN]
```

The following example shows how to create a library called MYLIB.

```
Library - MC68000 Library Utility    01-Dec-82
(c) Copyright 1982 Silicon Valley Software, Inc.

Listing file - [RETURN]
Output file [.obj] - MYLIB
Input file [.obj] - BIGUNIT
Input file [.obj] - SPEC.UNIT
Input file [.obj] - SNDUNIT
Input file [.obj] - LINEUNIT
Input file [.obj] - [RETURN]
```

To use an EXEC file to build a library, you must create a text file that has a call to the Library followed by file names. The following example shows how you could build MYLIB using an EXEC file.

First, create and save a text file with the following information in the following order:

```
LINKER < %

MYLIB
BIGUNIT
SPEC.UNIT
SNDUNIT
LINEUNIT
```

If we save this file with the name LIBBUILD.TEXT, we may use it in the following manner:

```
Press   [ExecFile]
Type    LIBBUILD
Press   [RETURN]
```

PROGRAM SPACE REQUIREMENTS

The line dividing code space and data space is known as the initial system stack pointer. The initial system stack pointer may be adjusted to accommodate software requiring more code space or more data space. Adjusting the initial system stack pointer reinitializes (reboots) the system. Drivers are loaded at the new initial system stack pointer, and volumes are mounted again.

Memory layout on the old 1.0c (Nov. 12, 1982) and the new 1.1a (Feb. 18, 1983) system is as follows:

memory size	screen	code	data	default sp
256K (old)	56K	136K	64K	9E000
256K (new)	56K	143K	57K	9C400
512K (old)	56K	328K	128K	AE000
512K (new)	56K	200K	256K	CE000

Space Requirements

The Pascal compiler requires 82K - 83K of code space, if it is
not segmented. If the pascal compiler is not segmented, the initial system stack pointer must be
readjusted to run the Pascal compiler on the 256K system. The command **SP 9D000** adjusts the system stack pointer to allocate sufficient code space to run the Pascal compiler.

The Fortran compiler requires a 512K machine.

Setting the Initial System Stack Pointer

The command **SP** stands for initial system Stack Pointer. This command is used to determine where the stack pointer is to be located in system memory. There are two forms of the **SP** command.

The **SP** command without arguments displays the current setting of the stack pointer:

```
Type   SP
Press  [RETURN]
```

The following displays in the Command window:

```
sp = 0009C400
```

The **SP** command with a parameter sets the initial stack pointer to that value if the parameter is valid. The parameter is interpreted as a hexadecimal number.

```
Type   SP 9D000
Press  [RETURN]
```

After the **SP** command sets the system stack pointer, it restarts the system. The following displays in the Command window:

```
Restarting ....
```

The operating system reinitializes by loading drivers, mounting volumes, etc.

The **SP** command can be issued from only the Dispatcher level. Any attempt to change the value of the initial stack pointer from a nested program results in an error message.

An attempt to set the stack pointer to an invalid value, such as an odd address or overlaying code and stack, results in an error message. In this case, the initial stack pointer value is not changed.

To test whether programs written on a 512K machine work on a 256K machine, you may reboot your 512K system as if it were a a 256K system as follows.

```
     Type   SP 9C400 256
     Press  [RETURN]
```

To set the stack pointer in a startup file, you must specify and additional parameter as follows:

```
     SP XXXXXX STARTUP
```

Example:

```
     SP 9D000 STARTUP
```

BUILDING ACTION TABLES AND ALTERNATE CHARACTER TRANSLATION TABLES

There are two new programs on the FSYSGEN distribution diskette, and can be found in the volume /CCUTIL after the system update has been performed. They are BLDACT and BLDALT. These previously mysterious programs provide you with far more control of your printers than was possible before now.

BLDACT is short for **Build Action Tables.** Action tables allow you to set the communication parameters necessary on a particular printer to perform enhancements such as underline, **bold face**, superscript, subscript and microspace justification of text. All these capabilities can now be accomplished provided the printer in use supports these features.

BLDALT is short for **Build Alternate Character Translation Tables.** An alternate character translation table allows you to create strings that will be sent to the printer to represent an alternate key on the Concept. The alternate characters are created by holding down the [ALT] key, which sets the Most Significant Bit. These are the upper 128 characters found in the Character set table in chapter 5 of the Personal Workstation User Guide.

A BLDALT table sends a character or string of characters for the printer to type when one an alternate character is encountered. For example, if you wanted the [ALT]-S code D3 hex to represent S you would give it the instruction "S" BACKSPACE "T" and it would print the T over the S. A more common use of BLDALT is for use with a different type of print wheel. A Spanish or German print wheel will have characters or symbols that are represented by a simple hex value. The English print wheel may have a standard character like a backslash in its place. If your display character set has a German character in an alternate key position the print wheel may recognize that character by the same code as the English backslash. You would want to send the backslash value to the printer. This is where the BLDALT table comes in handy.

Using BLDACT is very simple, and you really don't need to have a great knowledge of escape codes to use it. The values that need to be included for the table can be found in your printer manual. If not, consult the dealer from which you purchased the printer.

A little more good news: if you are using the C-ITOH or Diablo printers supported by Corvus, there are already Action Tables made and on the FSYSGEN diskette. These files begin with the letters PAT (for printer action tables), and can be looked at for examples of how to build other tables. The other files that were included on the distribution diskette are CITOH.TEXT and DIABLO630.TEXT which are Exec files that will load the printer driver and Action Tables for the C-ITOH and DIABLO 630 printers.

They can be executed by pressing [ExecFile] and typing the filename, or by placing the line in the STARTUP.TEXT file.

To use BLDACT simply type /CCUTIL/BLDACT. The entire table displays and is empty. To see how a table looks when completed load the file PAT.DIABLO630 from the FSYSGEN diskette. To do this press the F6 key [ReadFile], and enter the file name. The table becomes filled.

Below is an empty printer action table:

```
---------------------------------------------------------------------
 BLDACT [1.0b]: Build Printer Action Table
 (C) Copyright 1983 Corvus Systems, Inc.
 -----------------------------------------------------

 Character sequences to perform :

         Underline On........
         Underline Off.......
         Bold On.............
         Bold Off............
         Reverse Line Feed...
         Back Space..........

 Subscript and Superscript control sequences

 6 LPI :  Normal Form Advance Distance......
          Sub and Superscript Form Advance..

 8 LPI :  Normal Form Advance Distance......
          Sub and Superscript Form Advance..

 Proportional spacing control sequences

 10 CPI :  Normal character spacing.........
           Plus 1/120th of an inch spacing..
           Plus 2/120th of an inch spacing..
           Plus 3/120th of an inch spacing..

 12 CPI :  Normal character spacing.........
           Plus 1/120th of an inch spacing..
           Plus 2/120th of an inch spacing..
           Plus 3/120th of an inch spacing..

---------------------------------------------------------------------
+-------------------------------------------------------------------+
|                                                                   |
|Enter character or select function.                                |
+-------------------------------------------------------------------+
   F1       F2        F4        F5         F6        F8        F10
+--------+--------++--------+--------+ +--------++--------++--------+
|        |        ||        |        | |        ||        ||        |
|Prev    |Next    ||Del Char|ClrField| |ReadFile||WritFile||Exit    |
+--------+--------++--------+--------+ +--------++--------++--------+
```

To use the BLDACT program try moving the cursor from place to place with the cursor keys. As you can see the cursor takes up the entire position of the word or control character it is working on. Using the different function keys you may delete characters, clear the field, read or write the file, or move to the previous or next character as you may with a cursor key. To enter a control character simply press the character or the appropriate key. For BACKSPACE press [BACKSPACE] do not type out the letters of the word. To move around in the file use the cursor keys, pressing [RETURN] will enter [RETURN] as a control character. Once again, the contents for the table should all be listed in your printer manual.

Using BLDALT is a little more complicated than BLDACT. BLDALT sets up alternate character set translation tables. These tables tell what series of actions the printer will take when it encounters an alternate character code. Alternate characters are discussed in the Personal Workstation User Guide under the chapter on character sets.

To run the BLDALT program type /CCUTIL/BLDALT. The screen displays the following:

```
------------------------------------------------------------------
 BLDALT [1.0a]: Build Printer Alternate Character Translation Table
 (c) Copyright 1983 Corvus Systems, Inc.
 -------------------------------------------------------



 CURRENT ALTERNATE CHARACTER

  No Current Character



------------------------------------------------------------------
+----------------------------------------------------------------+
|                                                                |
|Enter Alternate Character or Press Function Key.                |
+----------------------------------------------------------------+
    F1       F2       F3        F5                  F6       F10
+--------+--------+--------++--------+         +--------++--------+
|        |        |        ||        |         |        ||        |
|ReadFile|Select  |ShowStrs||DletStr |         |WritFile||EXIT    |
+--------+--------+--------++--------+         +--------++--------+
```

At this point the program wants to have a file read in, or a character selected to begin a new file. Since there is no BLDALT table already made we are going to have to start from the beginning. Hold down the [ALT] key and select a character. For example, Hold down [ALT] and press S. The window now appears as follows:

```
----------------------------------------------------------------

 CURRENT ALTERNATE CHARACTER

  ALTERNATE "S"                 DECIMAL VALUE = 211
                                HEX VALUE     = $D3

----------------------------------------------------------------
```

To create a string of commands for the printer we must select this character, pressing it only makes it the current character. Press [Select], the F2 key. Now the window changes and a translation string can be created. The screen looks like this:

```
----------------------------------------------------------------

 CURRENT TRANSLATION STRING
   No string for this character.

 Enter new string for Current Character:

 _

----------------------------------------------------------------
```

Any key pressed now becomes a part of the string. To use the example shown earlier, S. press S, press [BACKSPACE], and press T. The string displayed should appear "S" BACKSPACE "T" unless something was typed in wrong. If you made a typo it cannot be corrected without typing the whole string in again. When a string has been entered press [RETURN]. The screen displays a prompt that asks "Do you want to save this translation string? [Y/N] N" The default is no, so if you do want to save the string press Y and [RETURN]. If you did not type it in correctly press [RETURN], the string will not be saved, and press [Select] again. The same character [ALT]-S is still the current character, and can have the proper string entered for it.

Now create a few other characters. Enter an alternate character on the keyboard, and then Select that character. Create a string, save it and create another. To see the list of strings that you have created press [ShowStrs]. The screen displays the following:

```
------------------------------------------------------------------

 List of Strings 1 to 2 in Table

    STRING 1 :  "S" BACKSPACE "T"
    Character codes : $D3

    STRING 2 :  "["
    Character codes : $C1

    STRING 3 :  ""
    Character codes : $CF

------------------------------------------------------------------
```

Strings 1 is really not important and I do not want to keep. There is another function called [DletStr] and, as you can probably guess, it can take care of unwanted strings. To return to the function key labels press [RETURN]. Press [DletStr] and the list of strings reappears. The Command window has the following message:

```
+-----------------------------------------------------------------+
|Delete String: Press [ESC] to stop, -1 to advance display        |
|                                                                 |
+-----------------------------------------------------------------+
```

Just above it in the System window is the prompt for what action you would like to take. "Enter string number to delete: -1" To delete string 1 press the number and press [RETURN]. If you press 1, [RETURN] you will be prompted with:

```
------------------------------------------------------------------

    STRING 1 :  "S" BACKSPACE "T"
    Delete this string? [Y/N] N

------------------------------------------------------------------
```

Press Y and [RETURN] to remove the string. You then have to press [RETURN] to leave the delete string level, press [RETURN]. Only one string may be deleted at one time, so to delete another string you must press [DletStr] and go through the same prompts again. Once you create the strings that you want to represent the [ALT] keys, you need to write the file.

Press [EXIT] to leave the program. There is a prompt asking if you want to save this table. The default is no, so press Y and [RETURN] to save it. Type in the volume name and filename you want it saved to and the program will create the file and save it.

In order to use the tables you have just created they must be loaded when you load your printer driver. After pressing [SetDatCm], and [PRINTER], press [PrtrFunc]. There are two keys here in particular [LdAction] and [LdAltChr] which load action tables and load alternate character translation tables. These functions must be performed in order to use your own specially designed tables.

By the way--we do have a few bugs to report in this package. If you plan to use a printer with anything other than 10 pitch (and write directly from EdWord) you will have to create a special Action Table. EdWord always uses the 10 CPI values from the table. The lines that define the pitch in the table are these:

```
 Proportional spacing control sequences

 10 CPI :  Normal character spacing.........
           Plus 1/120th of an inch spacing..
           Plus 2/120th of an inch spacing..
           Plus 3/120th of an inch spacing..

 12 CPI :  Normal character spacing.........
           Plus 1/120th of an inch spacing..
           Plus 2/120th of an inch spacing..
           Plus 3/120th of an inch spacing..
```

If you plan to use 12, or 15 pitch directly from EdWord make a separate table for 10, 12, and 15 pitch by placing the correct values for the selected pitch in the table for 10 CPI. Load the appropriate table each time you want to use the printer. For the 12 pitch table use the values given under the 12 CPI listing and place them in the 10 CPI section. To create a 15 pitch table set the 10 CPI values with printer data for 15 pitch.

The same type of problem also occurs with trying to print at 8 LPI. To print at 8 LPI a table must be created with the appropriate values for 8 LPI in the 6 LPI section. The lines that define the LPI are these:

```
 Subscript and Superscript control sequences

 6 LPI :  Normal Form Advance Distance......
          Sub and Superscript Form Advance..

 8 LPI :  Normal Form Advance Distance......
          Sub and Superscript Form Advance..
```

The third bug we have to report is a problem in creating an ESCAPE in the Alternate Character Translation Table. Whenever you press [ESC] a double escape, or ESCAPE ESCAPE, is entered in the table.

We hope these problems will not inconvenience you too much. By the time you are used to making the changes we hope to have the problem corrected.