OS-65U REFERENCE MANUAL V1.44

D3201/2-83

We encourage readers to submit corrections and suggestions for future editions of this manual. Ohio Scientific, Inc. may use or distribute the information you supply; you may, of course, also continue to use that information. Please address your comments to the attention of the Publications Manager:

Ohio Scientific, Inc.
Technical Publications Department
7 Oak Park
Bedford, MA. 01730

Order additional publications from your Ohio Scientific, Inc. dealer, or from:

Ohio Scientific, Inc.
Publications Distribution
1333 South Chillicothe Road
Aurora, Ohio 44202

TABLE OF CONTENTS

PREFACE

This guide describes the Ohio Scientific OS-65U operating system version 1.44. It is a guide for operating system users and has two primary purposes: that of a tutor for the new user and a reference manual for the experienced user. To introduce new users to the features of the operating system, the guide starts with a single-user system configuration. Depending upon the system purchased, this single-user environment can be based either on a floppy diskette and (a portion of) a hard disk or on two floppy diskettes. The instructions are written to work on either type of system. The manual then proceeds to a full hard disk configuration with the multi-user features of timesharing, networking and print spooling. This approach allows the new user to become familiar with the operating system in the simplest environment possible and at the same time provides a highly usable reference document for the experienced user. Care has been taken to arrange information in the guide in the order it will be needed by the programmer; this arrangement has the purpose of keeping the information relevant, minimizing the learning curve and, in general, simplifying the job of programming with OS-65U. Due to this arrangement of information, the operating system's features are 'unveiled' step-by-step as the guide proceeds. Therefore, it is recommended that the entire guide be read carefully before attempting to use the system. The temptation to 'read just enough to start programming' should be resisted; doing so could leave the user unaware of some of the system's most powerful and useful features.

It is presumed that the user of this guide has BASIC programming experience. However, a complete overview and summary tutorial of the language is provided; this includes a list of OS-65U BASIC commands and reserved words as developed by Microsoft with Ohio Scientific additions.

Those responsible for the operation and/or administration of the computer system should read this guide carefully, as well as those who will be using the system. In addition to the Guide, the user should refer to the following documents:

- Operation manuals for the terminal(s) and printer(s) purchased with the computer.

- User documentation for the system hardware.

- User documentation for the application programs purchased for the system.

Other recommended documents the user may require for additional information include:

- *Masterkey Series Field Service Manual,* a detailed description of the logic, circuitry, structure, and maintenance of the standard hardware used in all Masterkey series microcomputers.

- *Operation and Maintenance Manual for the 3300 Disc Drive,* a detailed installation, operation, and maintenance manual that covers all models of the fourteen-inch hard disk. (Published by Ohio Scientific Memory Products, Inc.)

- Operation manuals for options and peripheral equipment used with the system.

SYSTEM INTRODUCTION AND OVERVIEW

The Masterkey Series computer system, running under the Keyware OS-65U operating system is easy and cost-effective to use in software development, maintenance, modification and execution. The programming language of OS-65U is interpretive BASIC using the 6502 microprocessor. A number of Ohio Scientific extensions to BASIC have been added to make program writing and debugging more efficient. These extensions include:

Line Editor
Error Trapping
Extended INPUT and PRINT Statements
Common Variables Mode
Character-oriented Disk Access
Terminal Independence
FLAG Commands

Perhaps the most important aspect of Keyware OS-65U to the businessman is that it provides the capability to upgrade easily from a single-user to multi-user (timesharing or network) system. With OS-65U, it is possible to write software that will work both on single-user or multi-user systems. This capability to upgrade makes the Masterkey Series operating under Keyware OS-65U, an intelligent system choice for system and application programmers as well as the businessman requiring a 'turnkey' package.

This chapter lists the Keyware OS-65U system diskettes and associated hardware configurations, provides items of required information concerning output devices and provides an overview of the operating system and OS-65U system diskette. It also includes a brief but important discussion about saving files and making backup diskettes.

SYSTEM DISKETTES

System diskettes shipped with the system depend upon the hardware configuration. All Keyware OS-65U diskettes are listed with brief descriptions below. Check this list against your diskettes to be certain the proper ones have been received.

Diskette Labels

Software Label	
A. KEYWARE OS-65U Version 1.44 CD-7 FLOPPY DISKETTE BASED	For systems with 7 megabyte hard disk.
B. KEYWARE OS-65U Version 1.44 CD-36/74 FLOPPY DISKETTE BASED	For systems with 36 or 74 megabyte hard disk.
C. KEYWARE OS-65U Version 1.44 CD-7 HARD DISK BASED	For systems with 7 megabyte hard disk
D. KEYWARE OS-65U Version 1.44 CD-36/74 HARD DISK BASED	For systems with 36 or 74 megabyte hard disk.
E. KEYWARE OS-65U Version 1.44 TIMESHARE -- FILES ONLY W/MPUTIL	Contains timesharing program files only; MPUTIL memory and test boots.
F. KEYWARE OS-65U Version 1.44 NETWORK -- FILES ONLY	Contains networking program files only; no system.
G. KEYWARE OS-65U Version 1.44 DISK SYSTEMS MANAGER	Operates with all hard disk systems.

The networking diskette (F) is available as an option for hard disk based systems.

"Floppy Diskette Based" means that the Keyware OS-65U system resident on the diskette is to remain on the diskette and the computer is to be bootstrapped from the floppy diskette (a "D" will be entered for the bootstrap command).

"Hard Disk Based" means that the Keyware OS-65U system resident on the floppy diskette must be installed onto the systems portion of the proper model CD-7, 36 or 74 hard disk unit. The computer will then be bootstrapped from hard disk (an "H" will be entered for the bootstrap command).

"Files Only" means the diskette does not contain an operating system but contains program files to be used with the operating system diskette.

The date on the diskette label is the physical duplication date of the diskette, not the release date of the version. The release date and version number will be displayed on the system console upon system boot-up.

The term 'Bootstrap' is used in this manual to mean the automatic process of retrieving the operating system kernel from Track 0 of the diskette, reading the kernel into memory and passing control to the starting point of the operating system.

SYSTEM DISKETTE FORMAT

Keyware OS-65U software occupies approximately 265000 bytes of a single-density 275000 byte diskette. The diskette is divided into a operating system section and a files section. The operating system section uses the first 25088 bytes of the diskette. When the diskette is bootstrapped, this section of the diskette is copied into the computer's memory and executed.

The files section of the diskette starts at address 25088 and extends nearly to the end of the diskette. This portion contains the 31 OS-65U programs files. The first file is always the directory file, "DIREC*" and is a required file. It is discussed in Chapter Two.

The second file on a system diskette is the program file "BEXEC*". This program is essential to any operating system in order for it to be bootstrapped. Diskettes that will not be bootstrapped do not require this program.

In summary, a floppy must have a system copied to it and have the files "DIREC*" and "BEXEC*" before it can be successfully bootstrapped. For storage diskettes, only the DIREC* file is required.

PRINT DEVICES

The printing device(s) used with the Masterkey series must be connected properly and the output address selected by the print program. For more information on this, refer to *Masterkey User Guide,* under "Printer Checkout". The Masterkey Series is equipped with the following four types of interface connectors which are associated with the most often used printer ports:

Devices #3,#8	RS-232 with Handshaking (Serial C)
Device #5	8-bit Parallel (Centronics)
Device #6	12-bit Parallel (Diablo/NEC)

Before performing any special configurations with any printing or other output devices, read the relevant Chapter Eight sections, "CONFIGURE SYSTEM PRINTER (PRTMAP)", "DEVICES 3 & 8 BASE ADDRESS PORT SET (PRTSET)", "PARALLEL WORD PROCESSING DRIVER (WPDRIV)".

KEYWARE OS-65U OVERVIEW

The Keyware 65U software consists of a 24K operating system based on an extended BASIC interpreter and 31 utility programs.

The Operating System

The Keyware OS-65U operating system consists of the following:

4K - System buffer

1K - Reserved space for Editor

19K - Extended BASIC interpreter

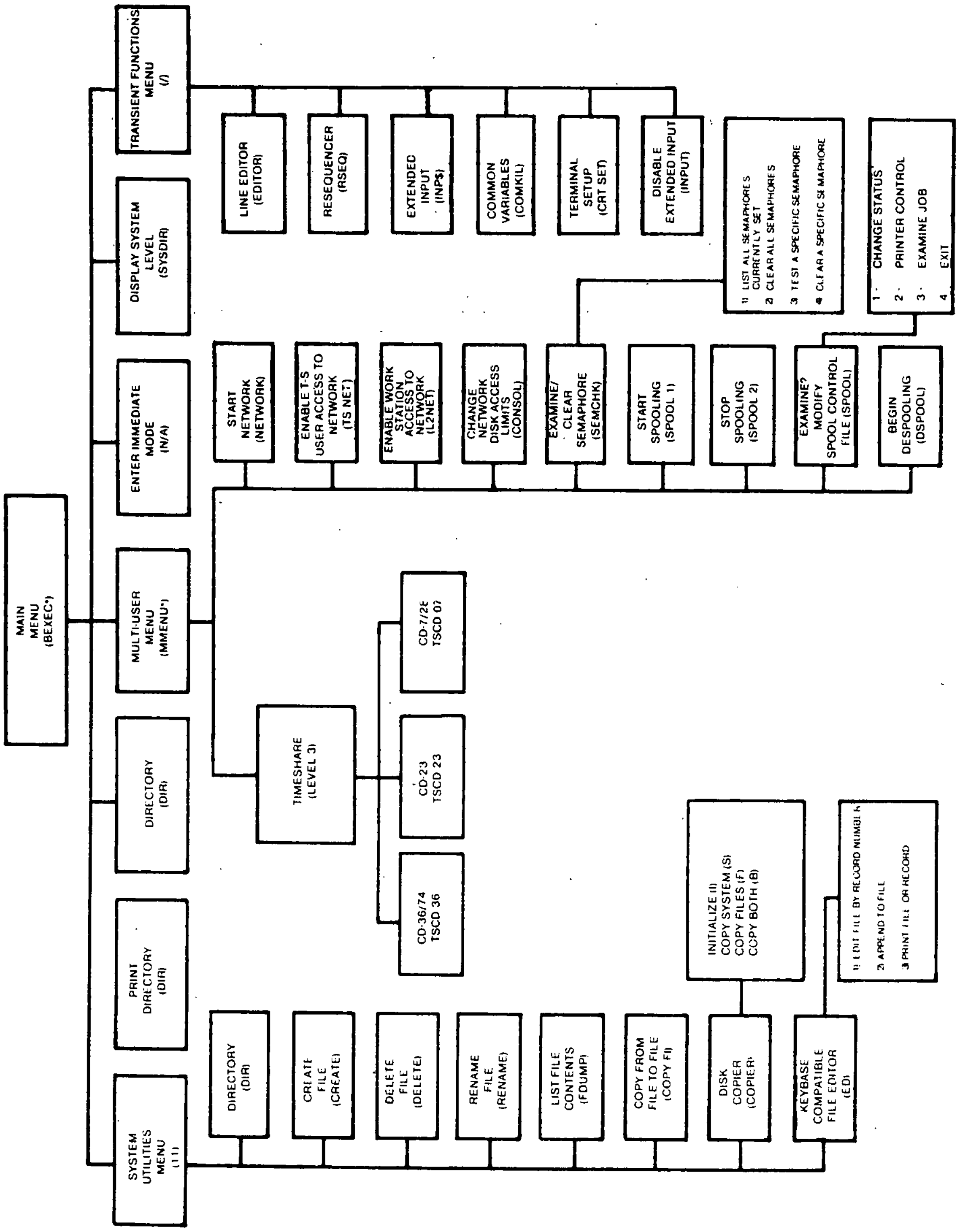
MAIN MENU (BEXEC*)
TRANSIENT FUNCTIONS MENU (/)
DISPLAY SYSTEM LEVEL (SYSDIR)
ENTER IMMEDIATE MODE (N/A)
MULTI-USER MENU (MMENU*)
DIRECTORY (DIR)
PRINT DIRECTORY (DIR)
SYSTEM UTILITIES MENU (11)
LINE EDITOR (EDITOR)
RESEQUENCER (RSEQ)
EXTENDED INPUT (INP$)
COMMON VARIABLES (COMKIL)
TERMINAL SETUP (CRT SET)
DISABLE EXTENDED INPUT (INPUT)
1) LIST ALL SEMAPHORES CURRENTLY SET
2 CLEAR ALL SEMAPHORES
3 TEST A SPECIFIC SEMAPHORE
4 CLEAR A SPECIFIC SEMAPHORE
1 · CHANGE STATUS
2 · PRINTER CONTROL
3 · EXAMINE JOB
4 · EXIT
START NETWORK (NETWORK)
ENABLE T-S USER ACCESS TO NETWORK (TS NET)
ENABLE WORK STATION ACCESS TO NETWORK (L2NET)
CHANGE NETWORK DISK ACCESS LIMITS (CONSOL)
EXAMINE/ CLEAR SEMAPHORE (SEMCHK)
START SPOOLING (SPOOL 1)
STOP SPOOLING (SPOOL 2)
EXAMINE/ MODIFY SPOOL CONTROL FILE (SPOOL)
BEGIN DESPOOLING (DSPOOL)
TIMESHARE (LEVEL 3)
CD-7/28 TSCD 07
CD-23 TSCD 23
CD-36/74 TSCD 36
INITIALIZE (I) COPY SYSTEM (S) COPY FILES (F) COPY BOTH (B)
1) EDIT FILE BY RECORD NUMBER
2) APPEND TO FILE
3) PRINT FILE OR RECORD
DIRECTORY (DIR)
CREATE FILE (CREATE)
DELETE FILE (DELETE)
RENAME FILE (RENAME)
LIST FILE CONTENTS (FDUMP)
COPY FROM FILE TO FILE (COPY FI)
DISK COPIER (COPIER)
KEYBASE COMPATIBLE FILE EDITOR (ED)

Figure 1-1.

The primary use of the system buffer and the BASIC extensions is for file input/output.

Immediate Mode and The Program Files

From the user's point of view, Keyware OS-65U may be divided functionally into two categories: immediate mode and the menu system. The user is either in immediate mode or within an OS-65U program.

Immediate Mode: Besides acting as the programming/debugging mode of the operating system, immediate mode communicates directly with the BASIC interpreter, allowing direct entry into and listing of any OS-65U utility program. All OS-65U programs are listed in Appendix A. Also, almost all BASIC commands, including OSI extensions, may be invoked from immediate mode. The means of entering and exiting immediate mode are covered in Chapter Two of this Guide.

The Menu System: Keyware OS-65U has menu-driven user interface software provided with the operating system. It is structured in a hierarchical (top-down) structure that begins with the Main, or BEXEC*, menu. As Figure 1-1 illustrates, the Main menu points to three other menus, each covering a major segment of programs -- the System Utilities, Transient Functions and Multi-User Utilities. The programs accessible from these menus are executed by the use of submenus and prompts. All programs, menus, and prompts are described and discussed in detail in this Guide.

PLAYING SAFE

It is wise to utilize reasonable protective measures when working with any computer system. These measures are described briefly here. Disk management and backup procedures are also outlined in the Masterkey User Guide.

Saving Files

The computer's memory can be inadvertently cleared by a number of occurences, including power problems, static electricity and operator error. Therefore, it is a good idea to SAVE, or write, the contents of memory to disk frequently. How frequently this should be done depends to some extent upon what is being entered, but saving every 15 minutes or so, particularly while programming, is certainly within reason. Instructions for using the SAVE command are in Chapter Two.

Backing Up Diskettes

Diskettes are easily lost, scratched, damaged or written over --and they eventually wear out. If power to the computer is cut off for any reason while a diskette is in the drive, the data in the track can be destroyed. It is a good practice to back up diskettes regularly and often. Once a day is standard, but some situations dictate doing it more or less often than that.

Duplicate copies should be made of all Keyware system diskettes and the originals should be stored in a safe, climate-controlled location. The COPIER program is used to back up diskettes; its use is described in the Chapter Two.

Write-Protecting Diskettes

Diskettes can be write-protected to prevent their being written over and the information on them lost. The small diecut notch at the edge of the diskette cover establishes write protection. It is recommended that all important diskettes be protected in this way.

USING THE SYSTEMS UTILITIES

This chapter is a tutorial beginning with system startup and guides the user through the Keyware OS-65U utility programs and BASIC commands required to effectively manage files and diskettes. Instruction is provided in initializing/copying diskettes as well as creating, modifying and using files.

The utility programs covered in this chapter are all accessible from the System Utilities menu. This menu, like all of the system's menus, are efficient to use and their use is recommended as much as possible. Since the System Utilities can also be invoked from the BASIC immediate mode, the utility program file names have been included (in parentheses) in the section headings.

In order to include all the necessary tools for using files, four BASIC commands, RUN, LOAD, SAVE, LIST, have been included in the chapter; however, they are covered in greater detail in Chapter Three. Since BASIC commands must be entered from immediate mode, detailed instructions for using immediate mode are also given.

SYSTEM STARTUP

Masterkey Series computers may be divided into two categories: hard disk based and floppy diskette based. Hard disk based systems have a (formatted) 7, 36 or 74 megabyte Winchester drive and 275 kilobyte floppy drive. Floppy based systems have two floppy drives. The steps for starting up both kinds are outlined below. For a more detailed description of the system hardware, see the *Masterkey User Guide.* For a detailed description of starting up and using the terminals purchased with the system, refer to the terminal user guide.

1. Turn on the computer, terminal and printer. Power switches are generally in view on the front or rear of the equipment.

2. With hard disk systems, insert the Keyware OS-65U (CD-7, 36 or 74) floppy based diskette in Drive "A" and close. Floppy based computer systems will have the CD-36/74 floppy based diskette; it should be inserted in DEVice "A", the top or right drive.

CAUTION

Bootstrap only from the floppy based diskette. Attempting to boot from the hard disk based diskette will destroy it.

3. Depress the reset key on front of the computer.

4. When the "H/D/M?" message appears on the screen, enter a capital "D", indicating the system is to be bootstrapped from a FLOPPY DISKETTE BASED diskette.

5. The Main menu will appear on the screen as follows:

```
1.)  DIRECtory
2.)  Print DIRECtory
3.)  System Utilities
4.)  Transient Functions
5.)  Enter Immediate Mode
6.)  Select File System
7.)  Select Multi-User Menu
```

The sytem is now ready for use. It is recommended that new users work in a single-user environment while gaining familiarity with the system.

MASTERKEY DISK DEVICES

Throughout this chapter, reference is made to the proper OS-65U DEVice to execute on. The DEVice assignments refer to the hardware devices that hold the hard disk or floppy diskettes.

Masterkey OS-65U DEVices are named as follows. They are referred to by these names throughout this manual.

HARD DISK SYSTEMS -- 220E, 230E, 230I, 250I, 250J, 250II, 250JJ.

DEVice A	=	The floppy diskette.
DEVice E	=	The Winchester hard disk.
DEVice F	=	The second Winchester hard disk.

FLOPPY DISKETTE SYSTEMS -- 220C, 230C (and Keymate C-100)

DEVice A	=	The top or right floppy diskette.
DEVice B	=	The bottom or left floppy diskette.
DEVices C & D	=	Custom configured systems where DEV C uses the second side of the DEV A diskette and DEV D uses the second side of the DEV B diskette.

BASIC IMMEDIATE MODE

Since the four BASIC commands (RUN, LOAD, SAVE and LIST) discussed in this chapter must be invoked from immediate mode, this section provides a functional description of entering/exiting immediate mode.

Entering Immediate Mode

The standard method of entering immediate mode is to select "Enter Immediate Mode" from the Main Menu. The program will ask for the password which is

```
UNLOCK
```

OK will then appear on the screen.

The NEW Command: If a program resides in the workspace, any lines entered with a line number will be added to it. In order to create new programs, the workspace must be cleared by the NEW command. It has the form

```
NEW
```

This command should always be used before typing in a new program. When used within programs, it can be used for read protection in place of END commands.

Exiting Immediate Mode

As indicated earlier, the OS-65U user is always operating in immediate mode or from within an OS-65U program.

CAUTION

Note that upon leaving immediate mode, memory is cleared. Be certain the file in memory has been SAVEd to disk before doing so.

The SAVE command is described later in this chapter. The sequence of commands to leave immediate mode is

SYSTEM PROMPT	RESPONSE
OK	DEV"n"
OK	RUN"x"

where DEV"n" is the drive with the OS-65U system diskette (which at this point will probably be "A"). The RUN"x" command requires the file name of the menu desired upon leaving immediate mode. The options are:

Menu	File Name To Enter
Main Menu	BEXEC*
System Utilities Menu	//
Transient Functions Menu	/
Multi-user Menu	MMENU*

Example To enter the System Utilities Menu from immediate mode, enter the sequence:

```
DEV"A"
RUN"//"
```

It is also possible to RUN the DIRectory program directly from immediate mode. The procedure for doing so is discussed in the following section.

THE DISKETTE DIRECTORY (DIR)

The directory of a diskette is accessed by the DIR program, which examines the contents of the DIREC* file (discussed later in this chapter) and displays those contents in a tabular format. The file name, type, access rights, diskette address and length are listed, followed by two fields that state whether the file starts on a diskette sector boundary and whether the file length is a multiple of a sector length (3584). Sector lengths are 3584 bytes each.

The directory of a diskette can be obtained from the Main menu, the System Utilities menu, or from immediate mode. To access it from the menus, select "Directory". The program will prompt

```
DEVice?
```

Respond with the drive name of the diskette directory you wish to see. The directory will then be listed on the console.

Note that the menus also offer the option of printing out the directory. To do this, select "Print Directory" from either the Main or System Utilities menu. Be certain the printer is on line. The program will prompt

```
DEVice?
```

Following the response, the directory will be printed out.

A diskette directory can also be obtained from the immediate mode. This is accomplished by RUNning the DIR program. The following sample sequence would produce the directory for DEVice "B". This example assumes the OS-65U diskette is on DEVice "A". This means that when Immediate mode is entered (see system startup), you are on DEVice "A". If another DEVice (such as "B" or "E") has been specified since, DEVice "A" must be specified again, since the DIR program resides on the OS-65U diskette.

PROMPT	RESPONSE
OK	DEV"A
OK	RUN"DIR
DEVice?	B

Directory Example Following is the directory listing for the Keyware OS-65U diskette.

```
OS-65U FILE DIRECTORY FOR DEVICE A

Name      Type   Access  Address  Length  Sec Bnd  Sec Len

DIREC*    Other  None    25088    3584    Yes      Yes
BEXEC*    Basic  Read    28672    7168    Yes      Yes
DIR       Basic  Read    35840    7168    Yes      Yes
CREATE    Basic  Read    43008    10752   Yes      Yes
RENAME    Basic  Read    53760    7168    Yes      Yes
COPYFI    Basic  Read    60928    10752   Yes      Yes
COPIER    Basic  Read    71680    10752   Yes      Yes
FDUMP     Basic  Read    82432    17920   Yes      Yes
FPRINT    Basic  Read    100352   3584    Yes      Yes
ED        Basic  Read    103936   10752   Yes      Yes
DELETE    Basic  Read    114688   3584    Yes      Yes
//        Basic  Read    118272   3584    Yes      Yes
EDITOR    Basic  Read    121856   10752   Yes      Yes
CRTSET    Basic  Read    132608   7168    Yes      Yes
INP$      Basic  Read    139776   10752   Yes      Yes
COMKIL    Basic  Read    150528   7168    Yes      Yes
CRT 0     Basic  Read    157696   10752   Yes      Yes
INPOUT    Basic  Read    168448   7168    Yes      Yes
RSEQ      Basic  Read    175616   7168    Yes      Yes
/         Basic  Read    182784   3584    Yes      Yes
SYSDIR    Basic  Read    186368   3584    Yes      Yes
MMENU*    Basic  Read    189952   7168    Yes      Yes
WPDRIV    Basic  Read    197120   3584    Yes      Yes
PACKER    Basic  None    200704   10752   Yes      Yes
CHANGE    Basic  None    211456   7168    Yes      Yes
DEFLIS    Basic  None    218624   3584    Yes      Yes
DEFILE    Basic  Read    222208   3584    Yes      Yes
GETCRT    Basic  Read    225792   3584    Yes      Yes
PRTSET    Basic  Read    229376   7168    Yes      Yes
PRTMAP    Basic  Read    236544   7168    Yes      Yes
INSTAL    Basic  Read    243712   21504   Yes      Yes

10752 Bytes Free
3 Sector(s) Free

31 File(s) in use
0 File(s) deleted
31 Total file(s) defined of 223 possible
```

EXITING AN OS-65U PROGRAM

When a Keyware program is finished, a carriage return prompt is issued which allows you to exit the program and return to the menu. If you do not wish to wait until the program is finished, the program can be aborted with the command

ABORT

in response to any program prompt. The prompt

Enter a <Return> to continue?

then appears. At this point, you may return to the menu that contains the program you are running in by pressing the <Return> key. Otherwise, you have the option of going directly into immediate mode by entering.

STOP

Example After entering an OS-65U program, it could be exited in the following manner:

PROMPT	RESPONSE
Device?	E
File Name (Six characters max)?	ABORT
Enter <Return> to Continue?	STOP
OK	

You would then be in immediate mode. If the <Return>key had been pressed instead of entering STOP, you would have returned to the System Utilities menu.

INITIALIZING DISKS AND DISKETTES (COPIER)

To be usable by the operating system, a disk or diskette must be 'initialized' into the OS-65U format, that is, divided into sectors of 3584 bytes with headers, etc. This is accomplished by means of the COPIER program.

CAUTION

Initialization is a destructive action--any information existing on the diskette is erased. Be certain all valuable information on the diskette has been copied before initializing it.

Instructions in this section are for initializing disks and diskettes which are used both with hard disk based and floppy disk based systems. The hard disk instructions are for initializing a portion (1 megabyte) of the disk. Procedures for initializing the full disk, testing it, checking for defective sectors, installing multi-user software, etc. are described in Chapter Nine (Multi-User Part) of this guide.

Hard Disks Initializing hard disk systems requires one step for the Masterkey CD-7 system and two steps for the CD-28, 36 & 74 systems. The CD-36 & 74 systems must be configured with the DSKSET program before they can be initialized.

Configuring CD-28, 36 & 74 systems (SHSET and DSKSET): Before the CD-28, 36 or 74 hard disk systems can be initialized, the OS-65U Floppy based diskette must be configured to work with them. It is not necessary to run a configuration program for CD-7 systems. To configure the diskettes, the SHSET (CD-28) or DSKSET (CD-36, 74) program must be run, using the steps below. The programs are on the "Timeshare--Files Only" diskette, so some diskette switching is part of the procedure.

1. Bootstrap the system, using the CD-36/74 floppy based diskette according to the "System Startup" instructions earlier in this chapter.

2. Remove the floppy based diskette and insert the "Timeshare--Files Only" diskette.

3. Enter immediate mode (see instructions earlier in this chapter) and invoke the DSKSET program with the command

```
RUN"SHSET        (CD-28)
RUN"DSKSET       (CD-36, 74)
```

The program then issues the following warning and prompt:

```
                    Warning

You are running a (CD-7 or 36) operating system.

Please make sure you have read the manual
before using this utility!!!

Do you wish to continue ?
```

4. Remove the Timeshare diskette and re-insert the floppy based diskette. Respond to the program prompts as follows:

PROMPT	RESPONSE
Do you wish to continue?	Y
Password?	DISC
DEVice?	A
(Please Wait)	
(SHSET) 1) Set Diskette to CD-28 Type 2) Set Diskette to CD-7 Type	1 or 2
(DSKSET) 1) Set Diskette to CD-74 Type 2) Set Diskette to CD-36 Type 3) Set Diskette to Floppy Only Type?	1 or 2
(Please Wait)	
Enter a <cr> to continue ?	

The changes take effect when the system is rebooted. The diskette is now configured for the specified hard disk. A carriage return will return you to the System Utilities menu.

CAUTION

DSKSET modifies the operating system. If a Read or Write error occurs during the execution of DSKSET, the state of the system on the diskette is unknown and should be considered defective. If this occurs, contact your dealer or M/A-COM OSI for assistance.

CD-38/74 hard disk based diskettes also must be configured with DSKSET. Configure it at this time, using the procedure described above.

Initializing The Disk: After the CD-36/74 floppy based diskette has been configured with DSKSET (not required for the CD-7 disk), the hard disk may be initialized. Procedures for initializing only a portion of the disk are given at this point; initializing it fully is not advised at this time. The procedure for initializing 1 megabyte of the disk is given below. 1 megabyte provides ample room forinstalling the single-user OS-65U software and creating scratch files to practice programming. The procedures for initializing the disk fully are given in Chapter Nine (Multi-User Part).

CAUTION

Some dealers install proprietary software on the hard disk; this could be erased when the disk is initialized. Check with your dealer before initializing any portion of the hard disk.

Use the following steps to initialize 1 megabyte of the hard disk.

1. Be sure the system is turned on. Insert the "Floppy Based" diskette and boot up according to the "System Startup" instructions given earlier in this chapter.

2. From the Main menu displayed on the screen, select "System Utilities".

3. From the System Utilities menu (below), select "Disk Copier" by number.

```
1) DIRECtory
2) Print DIRECtory
3) Create File
4) Delete File
5) Rename File
```

6) Dump File Contents
7) Copy From File to File
8) Disk Copier
9) General purpose DMS+ compatible file editor

4. From the Disk Copier menu (below), enter "I" for Initialize.

```
Initialize      (I)
Copy System     (S)
Copy Files      (F)
Copy Both       (B)
Exit <cr>?
```

5. Respond as follows to the prompts:

PROMPT	RESPONSE
DEVice?	E
Password	1000 (CD-7) 3300 (CD-36/74)
From address?	0
To address?	1000000
Will initialize 0 through 1003519. Alright?	Y
Wait 1.2 minutes	

1 megabyte of the disk will then be initialized as specified. When finished, the Disk Copier menu will reappear.

Floppy Diskettes

The procedure description for initializing floppy diskettes depends to a certain extent upon whether the diskette is initialized on a hard disk system or a floppy diskette system. This is because two disk DEVices are needed; Masterkey Series hard disk systems use their one floppy drive (DEVice A) and the hard disk (DEVice E). Floppy diskette systems use their two floppy drives (DEVices A & B). All variances are noted in the procedure description. This description assumes that the hard disk (DEVice E) has been initialized but that the hard disk software has not been copied onto it as yet.

With the power on, insert the OS-65U floppy based diskette in DEVice A and bootstrap according to the system startup instructions given earlier in this chapter. From the Main menu, select "System Utilities". From the System Utilities menu, select "Disk Copier". The Disk Copier menu (below) will appear.

```
Initialize      (I)
Copy System     (S)
Copy Files      (F)
Copy Both       (B)
Exit <cr>?
```

Select "Initialize (I)" from this menu.

At this point, the diskette to be initialized should be inserted into the drive. Hard disk system users should remove the OS-65U floppy based diskette from DEVice A and insert the blank diskette there. Floppy diskette system users should insert the blank diskette into DEVice B.

The program prompts for the necessary information to initialize the diskette. Respond to the prompts as follows:

PROMPT	RESPONSE
DEVice?	A (Hard Disk System) B (Diskette System)
From address?	0
To address?	275967

The program will take a few moments to initialize the diskette; it will then return to the System Utilities menu. The number 275967 will initialize the entire diskette. It is possible to respond to the "To address?" prompt with a number close to 275967, such as 275000. The program will then prompt

```
Will initialize to 275967. Alright? (Y/N)
```

to which the appropriate response should be given.

COPYING DISKETTES (COPIER)

The COPIER program is used to copy entire diskettes -- or major portions of them. One of its primary uses is for making backup copies of diskettes. In fact, it is advisable to use the procedures described below to make backup copies of all of the Keyware OS-65U diskettes shipped with the computer system. The procedure description for copying diskettes varies somewhat, depending on whether it is done on a hard disk or floppy diskette system. The points of variance are noted.

Before a diskette can be used as a destination diskette, note that it must have been initialized by OS-65U. Instructions for initializing diskettes are in an earlier section of this chapter.

To invoke COPIER, select "Disk Copier" from the System Utilities menu. The Disk Copier menu will appear as follows:

```
Initialize      (I)
Copy System     (S)
Copy Files      (F)
Copy Both       (B)
Exit <cr>?
```

Besides initializing, the menu offers the option of copying the (operating) System portion of the diskette, the Files portion, or Both the System and File portions. Recall from the discussion of the OS-65U diskette in Chapter One of this manual, that a portion of the diskette has been reserved for the operating system and for the OS-65U (or other) files. The system portion is reserved for the operating system, whether it is copied there or not.

Copying Diskettes With Hard Disk Systems

Copying diskettes with a Masterkey Series hard disk system involves a 2 step procedure: 1) Copying the diskette to the hard disk, and 2) Copying the contents back to another diskette. To provide a relevant example, the procedure for making copies of the OS-65U diskettes will be described; the same procedure works for any diskette to be copied. The steps are as follows:

1. Insert the OS-65U floppy based diskette in DEVice A and display the Disk Copier menu (see previous instructions).
2. Select (B)oth from the Disk Copier menu.
3. Respond to the program prompts as follows:

PROMPT	RESPONSE
From DEVice (<cr>=A) ?	A
To DEVice ? (<cr>=B) ?	E
To system based at address ?	0
Are you sure?	Y

The OS-65U diskette will then be copied one sector at a time to the hard disk. The Disk Copier menu will reappear when the copy is complete.

To copy the hard disk contents back to another diskette, use the following procedure:

1. Be certain the destination diskette has been initialized by OS-65U. Insert it in DEVice A.
2. Display the Disk Copier menu.
3. Select (B)oth from the menu; respond as follows to the prompts:

PROMPT	RESPONSE
From DEVice (<cr>=A)?	E
To DEVice (<cr>=B)?	A
From system based at address?	0
Are you sure?	Y

The floppy based diskette (contents of the hard disk) will then be copied to the destination diskette. Since the contents of the diskette are still on the hard disk, the last half of the above procedure can be used to create as many backup copies of the diskette as necessary.

The procedure described above may be used on a hard disk system to copy any diskette. Note, however, that after making backup copies of OS-65U diskettes, most diskettes being copied will not contain the operating system. If the system portion of the diskette is empty, use the "Copy Files" selection on the Disk Copier menu; attempts to use the "Both" selection will produce an error. Descriptions of the Disk Copier menu selections are covered next.

Copy System (S): Use this option to copy just the operating system portion of the diskette (addresses 0 -- 25087). The Keyware OS-65U operating system can always be safely copied to a disk since the area is reserved for them.

Be certain to use the proper DEVice names for the source and destination disk(ette) while using the procedures described above for copying the System portion of the disk(ette). If the system portion of the source disk(ette) is empty, a Read error will result at address 0.

Copy Files (F): Make this selection from the COPIER menu to copy just the files on the disk(ette) (addresses 25088 -- 275967). Follow the copying procedure above.

Copy Both (B): Use this selection for copying the entire disk. After issuing the sequence of prompts listed above, the program will copy the entire diskette. If either the system portion or files portion is not on the source disk(ette), a Read error will result. The missing portion will be indicated by the address where the Read error occurs (address 0 indicates the system portion, address 25088 the files portion).

Copying (Installing) the Hard Disk Based Diskette: It is recommended that hard disk system users create backup copies of the OS-65U hard disk based diskette at this time, using the "Both" selection procedure exactly as described above. Doing so will create backup copies of the diskette, and, of equal importance, copy the hard disk based diskette to the hard disk. When this is completed, the hard disk (DEVice E) may be used as the basis for all remaining operations described in this manual. Recall that approximately 1 megabyte (1 million bytes) of the disk has been initialized. The hard disk based software occupies 275967 bytes; the rest is available for storage.

Copying Diskettes on Floppy Based Systems

Users of floppy diskette systems use the COPIER program to copy entire diskettes or major portions of them. Before copying a diskette, be certain that the destination diskette has been initialized. To invoke COPIER, select "Disk Copier" from the System Utilities menu. The Disk Copier menu will appear as follows:

Initialize	(I)
Copy System	(S)
Copy Files	(F)
Copy Both	(B)
Exit <cr>?	

As a relevant example, the procedure will be described for making a copy of the OS-65U floppy based diskette. The same procedure can be used to copy any diskettes, but note that the "Copy Both" selection is used only when both a System and Files exist on the diskette; if either is missing, an error will result. Use of the Disk Copier menu selections are described below.

Copy System: Use this option to copy just the operating system portion of the diskette (addresses 0 -- 25087). The Keyware OS-65U operating system can always be safely copied to a disk since the area is reserved for them.

Be certain the source and destination diskettes are placed in the correct DEVices at this point. Normally, the source diskette replaces the OS-65U diskette in DEVice A and the destination diskette goes in DEVice B. Then respond to the prompts as follows:

PROMPT		RESPONSE
From DEVice	(<cr>=A)?	<cr>
To DEVice	(<cr>=B)?	<cr>
Are you sure?		Y

The program will then copy the system portion of the disk sector by sector and return to the COPIER menu. If the system portion of the source disk is empty, a Read error will result at address 0.

Copy Files: Make this selection from the COPIER menu to copy all files on the disk (addresses 25088 --275967). The program issues the same prompts as above and then copies the files portion of the disk one sector at a time.

Copy Both: Use this selection for copying the entire disk. After issuing the sequence of prompts listed above, the program will copy the entire diskette. If either the system portion or files portion is not resident on the source disk, a Read error will result. The missing portion will be indicated by the address where the Read error occurs (address 0 indicates the system portion, address 25088 the files portion).

CREATING FILES (CREATE)

After the storage diskette has been initialized, it is possible to create program or data files. Note that Keyware files must be created before they can be written to.

The first file that must be created on the diskette is the DIREC* file, since it acts as a catalog for the other files on the diskette. The DEVices vary somewhat in the procedure description, depending upon which disk(ette) the file is to be created on. The possibilities are:

Hard Disk Systems

The file is to be created on the hard disk (DEVice E).
The file is to be created on a floppy disk (DEVice A).

Floppy Diskette Systems (DEVice A or B)

Creating the DIREC* File

The DIREC* file is required on all diskettes. It is not a concern for the hard disk at this point, since the hard disk based software that has been copied to the disk included the DIREC* file. The CREATE program detects the absence of the DIREC* file and requires the user to create it before any other files can be created.

To create the DIREC* file with a hard disk system, insert an initialized blank disk in DEVice A. On floppy diskette systems, insert the Keyware OS-65U diskette in DEVice A and the initialized blank diskette in DEVice B.

On both types of systems, you must now reach the Main menu, either by returning from another menu or, by using the System Startup procedure. From the Main menu, select "System Utilities". From the System Utilities menu, select "Create File". The program will prompt for the information required to create the file, as illustrated by the creation of the following sample DIREC* file. Note that some of the responses (file name, file type, access rights, password) are automatically entered by the program.

PROMPT	RESPONSE
DEVice?	A (Hard Disk System) B (Floppy System)
No DIREXC* File/Continue?	Y
File Name (6 chars max)?	DIREC*
Max length in bytes (decimal)?	3584
File Type?	O
Access Rights?	N
Four letter Pass Word?	PASS

The maximum length of the file must be a multiple of 3584, which is the size of a sector. Each file name to be entered into the DIREC* file will require 16 bytes. For more diskettes, 3584 bytes will be sufficient. The maximum DIREC* file size possible is 65536 bytes; this maximum will be discussed later in the guide -- it only becomes a factor when fully initializing a hard disk.

After the responses have been entered, the program will display the DIREC* file parameters across the screen showing the file title, type, access, size and password as for our sample DIREC* file below.

```
DIREC*    OTHER    None    3584    PASS
```

The following prompts will then appear in sequence:

PROMPT	RESPONSE
Are you sure?	Y
Please Wait	
nnnnnn Bytes Free on Disk	

At this point, a menu appears that offers the choice of creating another file, looking at the diskette directory or returning to the System Utilities menu. Select the appropriate choice.

```
Another      (A)
DIRECtory    (D)
Exit <cr>?
```

An empty DIREC* file is now the sole resident of the storage diskette.

Creating Storage Files

The CREATE program is used to create storage files. After the "Create File" selection is made from the System Utilities menu, and the program detects that a DIREC* file exists, the following prompts are issued.

```
Device?
File Name (six characters max)?
Maximum length in bytes (decimal)?
 File Type
 Data (D)
 Basic (B)
 Other (O)

Access Rights
 None (N)
 Read (R)
 Write (W)
 Read/Write (RW)?

Four Letter Pass Word?
```

With the above entered, the file's parameters are displayed:

```
File Name   Type   Access Rights   Size   Password

Are you sure?

nnnnn Bytes Free on Disc.

Another      (A)
DIRECtory    (D)
Exit (cr)?
```

After the desired size of the file is entered, the program will choose the next higher multiple of 3584 as the actual size. Use forethought in planning the size of a file; it cannot be altered once the file is created. Should a file need to become larger, the result can be accomplished by creating another, larger file and copying the existing one to it. The procedure for copying files is covered later in this chapter.

Definitions of file types are:

Data	General data storage
BASIC	BASIC program file
Other	DIREC*, special function system files

Definitions of access rights are:

None	No access without password
Read	Read only without password
Write	Write and/or Run without password
Read/Write	Read and write without password

"Write" is used for "Run only" access.

When establishing a password, be certain to use one that can be easily remembered -- and record it somewhere; if it is forgotten, all access to the file will be limited to that of the access rights. With the proper password, all files have Read/Write access.

Example The steps below are used to create a BASIC program file "TEST01" on the hard disk:

```
PROMPT                                  RESPONSE

DEVice?                                 E
File Name (six characters max)?         TEST01
Maximum length in bytes (decimal)?      20000
File Type
 Data (D)
 Basic (B)
 Other (O)                              B

Access Rights
 None (N)
 Read (R)
 Write (W)
Read/Write (RW)?                        R

Four Letter Pass Word?                  JOHN

TEST01    JOHN    Basic    R    21504

Are you sure?                           Y

684543 Bytes Free on Disc.
```

```
Another        (A)
DIRECtory      (D)
Exit (cr)?
```

After the carriage return, the BASIC program file "TEST01" resides on the hard disk and the user is returned to the System Utilities menu.

LOADING FILES Loading files means to copy them from disk to memory. To do so requires the LOAD command from immediate mode with the following commands:

PROMPT	RESPONSE
OK	DEV"n"
OK	LOAD"File name","Password"

where "n" is the DEVice where the file is resident. Files with access rights of "Read" or "Read/Write" do not require a password. Files with "Write" or "None" require the password; failure to give it results in a message

```
?DEV n ERROR 130
```

If the program is unable to find the file named, it will respond with

```
?DEV n ERROR 128
```

(See Appendices for complete list of error codes). If you receive this error code, be certain the DEVice and file name have been entered correctly. If necessary, check the file names in the disk directory. The procedure for listing the directory is described earlier in this chapter.

Example To LOAD the "TEST01" program created earlier, use the following sequence:

```
NEW
DEV"E"
LOAD"TEST01"
```

OK will appear on the screen, indicating the program has been copied into memory. It may now be run, listed, edited, etc.

RUNNING PROGRAMS Running a program means to initiate the execution of a program that is on the disk or residing in memory. When RUNning a program on the storage diskette, the file will automatically be loaded into memory with the following commands:

```
DEV"n"
RUN "File Name","Password"
```

DEV"n" specifies the diskette containing the file. The password entry is not required to RUN files with Read or Read/Write access. See the Appendices for OS-65U system passwords.

Example The OS-65U program DIR existing on DEVice E is run with the following commands:

```
DEV"E"
RUN"DIR"
```

Since DIR has Read access, no password is required.

SAVING INFORMATION IN FILES

The BASIC command SAVE is used to copy files from memory to diskette. The advisability of using the SAVE command frequently is discussed in the introduction to this manual.

Invoking the SAVE program may be done in one of the two ways listed below:

```
SAVE
SAVE"File Name","Password"
```

To SAVE, files with access rights of "Read" or "None" require a password.

Before invoking SAVE, be sure to read and understand the following points:

1) SAVE is a destructive action; all information in the file is replaced.

2) If SAVE is invoked without a file name, the program will write the material to the last file loaded into memory. This feature is useful to the programmer, since it makes it possible to SAVE to the file without entering its name. Be certain, however, of which file was last loaded; writing information into the wrong file can be an unpleasant experience.

3) The destination file must have been previously CREATEd. If it has not been, the error message "DEVice n error 128" will appear.

4) If a program is larger than the file created for it, an error message results when the program is SAVEd; the program is held in memory. This situation could cause problems. Recall that you cannot leave immediate mode to CREATE a file at this point, since leaving immediate mode clears memory. To preclude this problem, CREATE a temporary storage file of sufficient size to hold any program likely to be written and keep it on the disk(ette). This temporary file will provide a place to store the program should the above situation occur.

LISTING FILE CONTENTS

Printing out the contents of a BASIC file or displaying it on a console requires the BASIC command LIST from immediate mode. The file must first be LOADed into memory. The LIST command has the forms:

```
LIST
LIST L
LIST#D
LIST#D,L
```

The first form, LIST, lists the entire program on the screen.

The second form, LIST L, is used to qualify the command as illustrated by the following examples:

LIST 10	lists only line 10
LIST -10	lists from beginning to line 10
LIST 10-	lists from line 10 to the end
LIST 10-20	lists from line 10 to line 20

The third form, LIST#D, directs the output to a printing device where "D" is the output device port number. The possibilities are 3, 5, 6, and 8. 3 and 8 are RS232 serial ports, 5 is a Centronics-type parallel line printer port and 6 is a Diablo-type 12-bit parallel Word Processing printer port. These are discussed in the *Masterkey User Guide*.

The last form, LIST#D,L, represents a combination of the previous two forms; it allows the user to specify the output device and the line numbers to print.

Note that it is possible to "page through" a program by stopping and restarting the LIST command. The listing can be stopped by <CTRL/S> and restarted by <CTRL/Q>.

Example The command

```
LIST#3,10-20
```

is required to print out program lines 10-20 on DEVice #3.

RENAMING FILES (RENAME)

Renaming a file begins at the System Utilities menu. Select "Rename File" from the menu; the RENAME program will then prompt for the necessary information in the following sequence:

```
DEVice?
Existing File Name?
Existing Password?
New File Name?
New Password?
New Access Rights (N,R,W,RW or <cr>)?

Are you sure?
```

If the file was created with Read/Write access, it has a password ANAN. In those cases, enter a period (.) as a response to the "Existing Password" prompt.

Example To rename the DEVice E file TEST01 to TEST02 requires the following sequence of instructions. Note that the password and access rights may also be changed.

PROMPT	RESPONSE
DEVice?	E
Existing File Name?	TEST01
Existing Password?	JOHN
New File Name?	TEST02
New Password?	1234
New Access Rights (N,R,W,RW or <cr>)?	<cr>
Are you sure?	Y

The file is then renamed. The <cr> (RETURN key) response to the "New Access Rights" prompt keeps the file's current access rights.

DELETING FILES (DELETE)

The program for deleting files is invoked from the System Utilities menu by selecting "Delete File". Note that the DELETE program only erases data in a file; it does not reclaim the space on the diskette for reuse. Deleted files are noted as such in the disk(ette) directory. Disk(ette) space is reclaimed by the PACKER program which is described in Chapter Eight of this guide.

Seleted space can be reused by the CREATE program -- if the file being created is the same size as the deleted file space.

The DELETE program prompts for information in the following sequence and then deletes the file:

DEVice?
File Name?
Password?

Are you sure?

If no password exists for the file (indicating R/W access), respond to the "Password?" prompt with a period.

Example Deleting the file TEST02 from the floppy drive on a hard disk system would require the following instructions:

PROMPT	RESPONSE
DEVice?	A
File Name?	TEST02
Password?	1234
Are you sure?	Y

TEST02 is then deleted from the diskette.

COPYING FILES (COPYFI)

The COPYFI program copies individual files to the same disk or diskette or to another one. To copy an entire set of files or an entire diskette, see the description of the COPIER program earlier in this chapter.

It is necessary to specify both the source file and destination file when copying files. Therefore, be certain the destination file has been CREATEd before using the COPYFI program. Creating files is described in an earlier section of this chapter.

The destination file must be as large or larger than the source file, even if the amount of data being copied will fit into a smaller file. The program will not allow a larger file to be copied to a smaller one.

Hard Disk Systems

COPYFI is invoked from the System Utilities menu. Select "Copy From File to File". If the file is to be copied from the hard disk (DEVice E) to a diskette in DEVice A, insert the destination diskette in DEVice A and go through the COPYFI prompts listed below. If the reverse is true, the file is to be copied from the diskette drive to the hard disk, insert the source diskette in DEVice A and go through the prompts below.

Floppy Diskette Systems

On a floppy diskette system, if a file is being copied to a file on another diskette, select "Copy from file to file" from the System Utilities menu. Remove the OS-65U diskette from DEVice A and insert the source diskette there. Insert the diskette with the destination file in DEVice B. Go through the COPYFI prompts below.

The COPYFI program prompts for information about the source file as follows:

```
From DEVice?
File Name?
Password?
```

It then prompts for information about the destination file:

```
To DEVice?
File Name?
Password?

Are you sure?
```

The file is then copied.

Recall that a period (.) is the correct response for the 'Password?' prompt if no password exists (R/W access files).

Example To copy file PROGA from the hard disk into a file PROGB on a diskette requires the following:

(Be certain the destination diskette has been initialized, contains a DIREC* file, and the PROGB file. It must be inserted in DEVice A.)

PROMPT	RESPONSE
From DEVice?	E
File Name?	PROGA
Password?	JOHN
To DEVice?	A
File Name?	PROGB
Password?	CARL
Are you sure?	Y

The file is then copied under its new name and password. Note that any new parameters such as name or password must have been specified when the file was CREATEd.

KEYWARE 65U BASIC COMMANDS AND RESERVED WORDS

As stated in the preface, users of this guide are presumed to be familiar with the programming language BASIC. Since Keyware OS-65U is a BASIC interpretive system, an in-depth discussion of the language is provided in this chapter to assist the programmer in understanding the system. Keyware BASIC was written by Microsoft, Inc. and enhanced by M/A-COM OSI. It is compatible with most other Microsoft BASICs.

This chapter is in a tutorial format; it provides a description of all commands and reserved words as well as a discussion of suggested programming techniques.

NOTATION

Special Characters

OS-65U BASIC makes use of special characters with fixed meanings. They are:

,	separates the elements of a list
;	separates the elements in a compressed output list
:	separates multiple statements on a line
.	used as a decimal point
?	can be used instead of PRINT
"	delimits a character string
( )	encloses array subscripts and determines the order of operations in arithmetic expressions
$	declares a variable to be a string variable
%	declares a variable to be an integer variable
space	a blank is ignored unless it appears in a character string enclosed in quotes
>	greater than
<	less than
+	addition
-	subtraction
*	multiplication
/	division
^	exponentiation

Keywords OS-65U BASIC contains words with fixed meanings called keywords or reserved words. They cannot be used as variable names. The following words are reserved:

ABS	FOR	NEXT	RUN
AND	FRE	NOT	SAVE
ASC	GOSUB	NULL	SGN
ATN	GOTO	ON	SIN
CHR$	IF	OPEN	SPC(
CLEAR	INDEX( )	OR	SQR
CLOSE	INDEX< >	PEEK	STEP
CONT	INPUT	POKE	STOP
COS	INT	POS	STR
DATA	LOAD	PRINT	TAB(
DEF	LEFT$	READ	TAN
DEV	LEN	REM	THEN
DIM	LET	RESTORE	TO
END	LIST	RETURN	USR
EXP	LOG	RIGHT$	VAL
FLAG	MID$	RND	WAIT
FN	NEW		

A quick reference to the use of these keywords is given in an Appendix.

ELEMENTS OF BASIC

Constants There are three types of constants in BASIC: numeric, string, and logical.

Numeric Constants: Numeric constants may have one of three forms: integer, real, and exponential. The integer form consists of a signed or unsigned string of decimal digits with no decimal point. Examples of the integer form would be 3 and -596. The real form has a decimal point. Examples of the real form would be 3.57 and -0.56. The exponential form is rEn where r is in real form and n is in integer form. The exponential form corresponds to scientific notation. For example, .00000156789 in exponential form would be 1.56789E-6 and 123778641 would be 1.23778641E+8.

All three forms of numeric constants are converted internally to the exponential form. The range of values is from -1.7014 E+38 to 1.7014 E+38 (2 to the power 127). The smallest positive value is 2 to the -128 power or 2.9387 E-39.

If a program calculates a value greater than 1.7014 E+38 an overflow error occurs. If a program calculates a value less than -1.7014 E+38 no underflow error occurs; a value of zero results.

Numeric constants have nine significant digits in the OS-65U BASICs. If more digits are used (not including the decimal point or exponent) the number is truncated. For example, 98765432111 is 9.87654321E+10.

String Constants: A string constant consists of a collection of up to 255 characters enclosed in quotes. Examples of string constants are "RATIO", "3.9", and "". Any character in the ASCII table except " (double quotes) may be used in a string constant.

Logical Constants: There are two logical constants in BASIC. They are TRUE and FALSE. TRUE is represented internally by -1 (all bits set) and FALSE by 0 (all bits reset).

Variables

A constant can be represented by a BASIC variable. There are three types of variables: numeric, integer, and string. Each type may be simple or subscripted. Simple variables are discussed in this chapter; subscripted variables are discussed later in this chapter.

Numeric Variables: A variable name consists of either a letter or a letter followed by another letter or decimal digit. Examples of numeric variables are A, XB, and S2.

A name with more than two letters can be used. Their use, however, should be watched closely because only the first two characters of a variable name are stored. Thus COVE and COUNT are considered to be the same variable (CO).

Integer Variablrs: Numbers can be stored in integer form in BASIC by adding a % (percent sign) to the variable name. Examples of valid integer variable names would be A , BA , and S2 . The range of an integer variable in -32767 to +32767.

String Variables: String variables are represented in BASIC by adding a $ (dollar sign) to the variable name. Examples of valid string variable names would be A$, BA$, and S2$.

Arithmetic Operators and Expressions

The arithmetic operators are represented by the following characters:

+	addition
-	subtraction
*	multiplication
/	division
^	exponentiation

An arithmetic expression calculates a numerical value. It consists of a list of variables or constants separated by arithmetic operators or brackets. Examples of arithmetic expressions are A+B, B+3, A+ 5, and (B ^ 2)-4*A*C.

Multiplication is never implied. Thus 3(AB) and (AB)(AC) are invalid and should be written as 3*(AB) and (AB)*(AC).

In order to avoid ambiguity in the evaluation of expressions a preference is established among the operators (see special character list earlier in this chapter). The preference is 1) parentheses, 2) exponentiation, 3) negation, 4) multiplication and division, and 5) addition and subtraction. Expressions enclosed in parentheses are performed first starting with the innermost pair of parentheses. All operations at one level are performed before proceeding to the next level. Thus

2 + 16 / 2 ^ 3   is   4

Operations at the same level are performed from left to right. Thus

2 ^ 3 ^ 2 is (2 ^ 3) ^ 2 or 64

It is recommended that the programmer insert parentheses in expressions to make them more readable and to ensure that the correct calculation is performed.

Bit Operators - AND, OR, NOT

The logical (or bit) operators AND, OR, and NOT operate bit-by-bit on the internal binary representations of the numbers. Each bit in the result is determined by comparing corresponding bits in the two numbers.

The operator AND sets a bit in the result to 1 if both corresponding bits in the numbers are 1. The operator OR sets a bit to 1 in the result if either one or both bits in the numbers are 1. The operator NOT operates on a single number and inverts the bits; each 0 becomes a 1 and vice versa.

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

OR

X	Y	X AND Y
1	1	1
1	0	1
0	1	1
0	0	0

NOT

X	NOT X
1	0
0	1

Some examples will serve to show how the logical operations work:

63 AND 16=16

```
    111111
AND 010000
----------
    010000
```

63=binary 111111
and 16=binary 100000
so 63 AND 16=16.

15 AND 14=14	15= binary 111 and 14=binary 1110 so 15 AND 14=binary 1110=14.
-1 AND 8=8	The two's complement representation of - 1 is 1111111111111111 2 (all bits of the two byte number set to 1). Therefore, -1=binary 1111111111111111 and 8= binary 1000, so -1 AND 8=8.
4 OR 2=6	4=binary 100 and 2=binary 10 so 4 OR 2=binary 110=6.
-1 OR -2=-1	=1 binary 1111111111111111 and -2 1111111111111110, so -1 OR -2=-1.
NOT 0=-1	The bit complement of sixteen zeros is sixteen ones, which is the two's complement representation of -1.
NOT X=-(X+1)	The two's complement of any number is the bit complement plus one.

A typical use of logical operations is "masking", testing a binary number for some predetermined pattern of bits. Such numbers might come from the computer's input ports and would then reflect the condition of some external device.

The action of the bit operators on the logical values -1 (TRUE) and 0 (FALSE) is the same as the familiar Boolean operators in the propositional calculus. This action is reviewed in the next section.

Relational Operators and Expressions

The relational operators are represented by the following characters:

=	equal
<	less than
>	greater than
<	less than or equal
>	greater than or equal
< >	not equal

Just as an arithmetic expression calculates a numerical value a relational expression calculates one of the two logical values TRUE or FALSE. Examples of relational expressions are:

A < = B	Is A less than or equal to B?
(A2 - B) < > 3	Is A2 minus B unequal to 3?

The bit operators AND, OR, and NOT can be used in relational expressions. If E1 and E2 are logical constants then

E1 AND E2	is TRUE only when both E1 and E2 are TRUE, otherwise FALSE
E1 OR E2	is TRUE when either one or both of E1 and E2 are TRUE, otherwise FALSE
NOT E1	is TRUE when E1 is FALSE, otherwise FALSE

Examples would be the following expressions which are in fact equivalent:

```
A < 0 OR A > 3

NOT (A >=0 AND A <= 3)
```

Relational operators may not be chained. The expression

```
1 < A < 5
```

is invalid and should be written

```
(1 < A) AND (A < 5)
```

The order of preference of arithmetic operators extends to relational operators and bit operators. The order of preference is 1) brackets, 2) exponentiation, 3) negation, 4) multiplication and division, 5) addition and subtraction, 6) all relational operators, 7) NOT, 8) AND, 9) OR. For example

```
NOT 2 * 3 > 5 is FALSE
```

Comparing Strings

Strings may be compared using relational operators. The comparison is made in the same manner as a dictionary ordering. Corresponding characters in the two strings are compared moving from left to right. One character is considered less than another if it precedes it in the ASCII table (see Appendix). Thus "ABE" is less than "ABF", "2" is greater than "12", and "$" is less than "%". An example of a relational expression involving strings is

```
NOT (A$ <=B$ AND C> 5)
```

String Expressions

String expressions consist of string constants, string functions, or string variables connected by the string operator . The result of a string expression is a string. The string operator = means concatenation.

For example,

```
10 A$="HELLO "
20 B$="THERE"
30 C$=A$ = B
40 PRINT C$$
```

results in the output:

```
HELLO THERE
```

ASSIGNMENT, INPUT AND OUTPUT

BASIC programs usually input data from the keyboard and output data to the screen. They can, however, communicate with a variety of other devices. Disk input and output are discussed later in this chapter.

LET Statement

The assignment of values to variables is performed by the LET statement. The forms are:

```
N   A = B
```

where N is a line number, A is a variable, and B is an expression. The Keyword LET should not be used. The expression B is evaluated and its value is assigned to A. Examples of valid assignment statements are:

```
10 LET X = 3.14159
20 A$ = "YES" + B
30 Z = 2 < 3
40 N% = 3.999
```

Numeric and integer variables may be assigned either numeric or logical values. If a numeric value is assigned to an integer variable, the fractional part is truncated. The value assigned to N% above is the integer 3. The value assigned to Z above is -1, representing the logical value TRUE. String variables may be assigned only string values. An attempt to assign a value to the wrong type of variable results in a type-mismatch error.

Chaining assignment statements as in the statement

```
10 A = B = C
```

will evaluate A to a logical value. The value of A will be -1 (TRUE) if B is equal to C, or 0 (FALSE) if B is not equal to C. So, the expression

```
10 A=B=C=0
```

would be evaluated from left to right as

```
10 A=((B=C)=0)
```

INPUT Statement

Input is obtained using the INPUT statement. The forms are:

```
N INPUT S
N INPUT "PROMPT";S
N INPUT#M,S
N INPUT%CH,S
```

where N is a line number, S is a list of variables separated by commas, and M is a device number.

An example of the first form is:

```
10 INPUT A,B
```

When this INPUT statement is executed, a question mark appears on the screen. Values are entered from the keyboard separated by commas. The second form of the INPUT statement allows a prompt to be printed along with the question mark. For example:

```
10 INPUT "YES OR NO";A$$
```

displays

```
YES OR NO?
```

on the screen. The response from the keyboard is assigned to the string variable A$.

The third form is discussed in this chapter under "Device Specified Input and Output". The fourth form is discussed in Chapter Four under "Data Channels".

READ and DATA Statements

The READ and DATA statements are always used together. The READ statements "read" the values in the DATA statements.

The DATA statement has the form

```
N DATA S
```

where N is a line number and S is a list of constants separated by commas. For example:

```
10 DATA 1.5, "HI", HI, -66
```

Strings may appear either quoted or unquoted. If unquoted, leading blanks are ignored and trailing blanks are included.

The values appearing in DATA statements are combined into a list in the order in which they appear. Thus the statement

```
10 DATA 2,3,5
```

is equivalent to the two statements

```
10 DATA 2
20 DATA 3
30 DATA 5
```

The READ statement has the form

N READ S

where N is a line number and S is a list of variables separated by commas. Each READ statement assigns values to the variables in its list by accessing the DATA list. The next READ statement proceeds in the DATA list where the previous READ statement left off. For example,

```
10 READ A,B$,C$
20 READ C
30 DATA 1.5,TYPE
40 DATA 40,50
```

is equivalent to

```
10 LET A=1.5
20 LET B$="TYPE"
30 LET C$="40"
40 LET C=50
```

Numeric values may be read into string variables. However, if an attempt is made to read a string into a numeric variable, a syntax error occurs in the line containing the string.

If there are more items in the DATA list than are read, the rest are ignored. On the other hand, if the DATA list contains too few items, then an out-of-data error occurs and the program is terminated.

RESTORE Statement

The RESTORE statement resets the pointer in the DATA list to the first DATA item. The RESTORE statement has the form:

N RESTORE

where N is a line number. For example:

```
10 DATA 10,20
20 READ A
30 READ B
```

assigns A the value 10 and B the value 20. While

```
10 DATA 10,20
20 READ A
30 RESTORE
40 READ B
```

assigns both A and B the value 10.

PRINT Statement

The PRINT statement is used for output. The forms are:

```
N PRINT S
N PRINT#M,S
```

where N is a line number, S is a list of expressions, and M is a device number.

The second form is discussed in this chapter under "Device Specified Input And Output".

The following example of a PRINT statement:

```
10 LET A=3.15
20 LET B$="TOTAL IS "
30 PRINT B$;A
```

results in

```
TOTAL IS 3.15
```

appearing on the screen.

A question mark can be used instead of PRINT when entering a program. The following examples are equivalent:

```
10 ?"THE VALUE IS ";B
10 PRINT"THE VALUE IS ";B
```

The question mark only appears when the program is first typed, and is replaced by PRINT when the program is listed.

Vertical Spacing of Output

Vertical spacing is accomplished by using the PRINT statement without an output list. This, in effect, prints a blank line. For example,

```
10 PRINT"LINE ONE"
20 PRINT
30 PRINT"LINE TWO"
```

results in the output:

```
LINE ONE

LINE TWO
```

Using a colon to allow multiple statements on a line and using ? for PRINT, three lines are skipped by this example:

```
10 ?:?:?:
```

Because the question mark is replaced by PRINT when the program is listed, the programmer should be careful of overrunning the end of a line if he uses a lot of question marks for PRINTS.

The following example will skip 32 lines or clear the screen.

```
10 FOR X=1 to 32:  PRINT:NEXT
```

Horizontal Spacing of Output

BASIC has several features that can be used to control horizontal spacing: zoned output, compressed output, the TAB function, the SPC function, and the POS function.

ZONED FORMAT: Each line of output is divided into 14-space zones. The use of commas in the output list specifies zoned format. For example,

```
10 PRINT"123456789012345678901234567890"
20 A=1.2:B=-5
30 PRINT A,B
40 PRINT A,,B
```

results in the output

```
123456789012345678901234567890
 1.2           -5
 1.2                         -5
```

All values are left-justified in their zones. Positive numerical values have a space in the first position instead of a plus sign.

If a PRINT statement ends with a comma, the next PRINT statement outputs to the next zone instead of the next line. The statement:

```
10 PRINT A,B
```

is equivalent to the two statements:

```
10 PRINT A,
20 PRINT B
```

If a value will not fit into the 14 spaces allowed for a zone (for example, a long string), it is extended into additional zones. The next value printed, if any, will be printed in the first zone not used by the preceding value.

Compressed Format

The use of a semicolon in the output list of a PRINT statement specifies compressed format. In the compressed format, output zones are not used. String values are printed next to each other. Numeric values are printed with a trailing blank. Positive numeric values also have a leading blank instead of a plus sign. For example,

```
10 B=-40:A=3.5
20 C$="THE ANSWERS "
30 D$="ARE "
40 E$=" AND "
50 PRINT C$;D$;A;E$;B
```

results in the output:

```
THE ANSWERS ARE 3.5 AND -40
```

If a PRINT statement ends with a semicolon, then the next statement outputs to the same line instead of the next line. For example,

```
10 PRINT A;
20 PRINT B
```

and

```
10 PRINT A;B
```

are equivalent.

TAB Function: The TAB function is used in the same way as the TAB key on a typewriter. The general form is

```
TAB(X)
```

where X is an arithmetic expression whose value is one less than the position where the next value is to be printed. An example:

```
10 PRINT A;TAB(3*X);B
```

Semicolons should be used with the TAB function. If followed by a comma, printing begins in the next zone. Note the effect of commas in lines 50 and 60 of this example:

```
10 PRINT"123456789012345678901234567890"
20 A=12.3:B=-5
30 A$="A"
40 PRINT A;TAB(8);B
50 PRINT A,TAB(8);B
60 PRINT A$,TAB(8),B
```

results in the output:

```
123456789012345678901234567890
 12.3    -5
 12.3                -5
A                                  -5
```

SPC Function: The SPC function is used to print spaces in output. The general form is

```
SPC(X)
```

where X is a numerical expression whose value is the number of spaces to be printed. For example,

```
10 PRINT"12345678901234567890"
20 PRINT"A";SPC(5);"B"
30 PRINT"A",SPC(5);"B"
```

results in the output:

```
12345678901234567890
A     B
A               B
```

Note that the comma in line 30 produces spacing within the zone.

POS Function: A PRINT statement can print a sequence of up to 132 characters in length. The position function returns (as an integer between 0 and 132) the position in the sequence of the last character printed. Its form is

```
POS(X)
```

where X is a dummy argument. The value of X is ignored. For example,

```
10 PRINT"01234";POS(X)
```

results in the output:

```
01234 5
```

There may be a difference between the position of a character in the output sequence and its position on the screen. This is because a video screen displays either 32 or 80 characters perline; the output sequence, which can be as long as 132 characters, may extend over several lines. Thus, when POS(X)=64 the cursor is at the right margin of the screen.

Device Specified Input and Output

OS-65U BASIC allows a device to be specified in PRINT, INPUT, and LIST statements. A device is specified by typing a pound sign followed by the device number. Some examples:

```
INPUT #8,D$
PRINT #4, "LINE PRINTER"
LIST #6
```

Input and output can be routed from or to various devices on the system. The following table lists the device numbers:

INPUT

1. Serial Console Port
2. Polled Keyboard
3. Serial Ports
4. Memory Input
5. Null
6. Null
7. Null
8. Serial Ports

OUTPUT

1. Serial Console Port
2. 540 Video
3. Serial Printer Ports RS232
4. Memory Output
5. Parallel Line Output
6. Null
7. Null
8. Serial Printer Ports RS232

must be taken not to route input or output to non-existent or turned-off peripheral devices since this will cause the computer system to "hang" and will require a reset which may destroy data in memory. For device numbers, please refer to Chapter 1 of this guide.

Money-Mode Output

Money-mode output of numeric variables is available in OS-65U BASIC. Any numeric variable output in the money-mode is automatically truncated to two digits after the decimal point. For example, 3.149 would be output as 3.14. Rounding up can be accomplished by adding .005555555 to the number to be output.

The money-mode also inherently provides left or right justification of the output in one of the 14-space output zones. A variable to be output in money-mode is preceded by either $R or $L, depending on whether it is to be left or right-justified in its field. Values are printed with a leading and following blank. When right-justified, values end two spaces inside the right edge of the field. For example:

```
10 X=1.429
20 Y=2.222
30 PRINT"123456789012345678901234567890"
40 PRINT $L,X
50 PRINT $R,Y
```

results in the output:

```
123456789012345678901234567890
 1.42
           2.22
```

BASIC turns on the money-mode when it encounters either $L or $R in an output list. The next numeric variable encountered in the output list is printed in money-mode, and then money-mode is turned off. String variables should not be used in money-mode because they do not turn it off.

PROGRAM CONTROL

Normally, program execution proceeds sequentially. The order of execution can be altered by the control statements described in this section.

GOTO Statement

The GOTO statement is an unconditional transfer statement. It has the form

```
N GOTO M
```

where N and M are line numbers. In OS-65U the directive M can also be a variable. Because blanks are ignored in BASIC, the command can also be written GO TO.

When the GOTO is executed, control transfers to line M rather than to the next statement. For example,

```
10 GOTO 30
20 PRINT "LINE 20"
30 PRINT "LINE 30"
```

results in the output

```
LINE 30
```

When used in the immediate mode, GO TO M starts execution of the program in the workspace at line M.

IF...GOTO Statement

IF...GOTO statement is a conditional transfer statement. It has the general form:

```
N IF X GOTO M
```

where N and M are line numbers and X is a relational or arithmetic expression. In OS-65U, the directive M can also be a variable. If the value of X is TRUE, then the next statement executed is line number M; if X is FALSE, control transfers to the line following N. For example,

```
100 IF A < = 5 GOTO 10
```

results in control passing to line 10 whenever A is less than or equal to 5 and to the line following 100 whenever A is greater than 5.

If X is an arithmetic expression, the value of X is treated as FALSE whenever X is zero. If the value of X is nonzero, then X is treated as TRUE.

The difference between statements and lines (which can contain several statements) becomes very important when using the IF ... GOTO statement. It should never be followed by a second statement on the same line; the second statement is never executed. Regardless of whether X is TRUE or FALSE, control always passes to a different line.

IF...THEN Statement

The IF...THEN statement is a conditional transfer statement. It occurs in two forms:

```
N IF E THEN M
N IF E THEN S
```

where N and M are line numbers, E is a relational or arithmetic expression and S is a statement. If E is an arithmetic expression, the value of E is treated as FALSE whenever E is zero. If the value of E is non-zero then E is treated as TRUE.

The first form is equivalent to the IF...GOTO statement described above. For example:

```
10 IF Z < > 0 THEN 300
```

and

```
10 IF Z THEN 300
```

both transfer control to line 300 whenever Z is non-zero.

With the second form, the statement R is executed whenever S is TRUE. If S is FALSE, R is ignored and control passes to the following line. These statements, for example,

```
10 INPUT "ENTER X";X
20 IF X > 0 THEN PRINT"X IS POSITIVE"
30 IF X<=0 THEN PRINT"X IS NOT POSITIVE"
```

will cause one, but not both, of the phrases "X IS POSITIVE" or "X IS NOT POSITIVE" to be printed.

Multiple statements can appear in the place of statement S. If S is FALSE, all the statements following THEN are ignored. For example,

```
IF Z > 0 THEN PRINT"Z IS POSITIVE":GOTO 300
```

prints "Z IS POSITIVE" and transfers control to line 300 when Z is positive. If Z is not positive, control passes to the next line in the program.

ON...GOTO Statement

The ON...GOTO statement is a conditional transfer statement having the general form:

```
N ON E GOTO L
```

where N is a line number, E is an arithmetic expression, and L is a list of line numbers separated by commas. The expression E is evaluated and truncated to an integer. Control then passes to the E-th line number in the list. For example,

```
20 ON Z GOTO 100,200,300
```

transfers control to line 100 if Z is 1, to line 200 if Z is 2, and line 300 if Z is 3. If Z is less than 1 or greater than 3 then the ON...GOTO statement is ignored and control passes to the following statement.

FOR-NEXT Statements

It is often desirable to repeat a segment of a program. Looping back over a portion of a program is usually accomplished in BASIC with a FOR-NEXT loop. The FOR and NEXT statements are used together to form the loop.

The FOR statement has the forms:

```
N FOR V = X TO Y
N FOR V = X TO Y STEP S
```

where N is a line number, V is a single numeric or integer variable, and X and Y and S are arithmetic expressions. The value of X is called the initial value assigned to the index variable V, the value of Y is the limit of V, and the value of S is the increment.

The NEXT statement has two forms:

```
N NEXT
N NEXT V
```

where N is a line number and V is the same index variable appearing in the FOR statement. The index variable is optional in the NEXT statement for a single loop and the inmost loop in a nested loop. It must appear for all other loops.

The FOR statement is the first statement in the program loop. The NEXT statement is the last statement in the loop. The collection of statements between the FOR statement and the NEXT statement is called the body of the loop and comprises the block of statements that is repeated.

The following actions take place with the first form of the FOR statement. When the FOR statement is executed, X, Y and S are evaluated, and the index is assigned the initial value. Then the body of the loop is executed. Two actions, increment and check, take place when the NEXT statement is executed. First, the index variable is incremented by adding 1 to its value. Second, the value of the index variable is now compared to the limit. If the value of the index variable exceeds the limit, control transfers to the statement following NEXT. If the index variable is less than or equal to the limit, control transfers to the first statement in the body of the loop. For example,

```
10 FOR I = 1 TO 5
20 PRINT I
30 NEXT
```

causes the numbers from 1 to 5 to be printed in a column.

Because the index is not compared to the upper limit until the end of the loop, the body is always executed at least once.

The expressions X, Y and S are evaluated only once, when the FOR statement is executed. Thus the looping is unaffected if the variables comprising these expressions are assigned new values within the body of the loop. Looping may be affected if the value of the index variable is changed within the body of the loop.

Control may transfer out of the body of the loop. Transferring into the body with a statement other than a RETURN from a GOSUB may cause errors or lead to other unexpected results.

STEP Clause: The second form of the FOR statement contains the STEP clause. In the first form, the index variable is incremented by 1 on each pass through the loop. In the second form, the index variable is incremented by the value of S. If S is positive, then control passes out of the loop to the statement following NEXT when the index variable exceeds the limit. If the increment is negative, control passes out of the loop when the index variable is less than the limit. If the increment is zero, no check is made and the loop repeats indefinitely. Consider these examples:

STATEMENT	VALUES OF X
FOR X=1 TO 2 STEP .5	1,1.5,2,2.5
FOR X=1 TO 5 STEP 10	1,10
FOR X=10 TO 1 STEP -1	10,9,8,7,6,5,4,3,2,1,0
FOR X=1 TO 10 STEP 0	1,1,1,...

Nested Loops

Loops may be nested. For example,

```
10 FOR I=1 TO 2
20 FOR J=1 TO 3
30 PRINT I,J
40 NEXT J
50 NEXT I
```

results in the output

```
1   1
1   2
1   3
2   1
2   2
2   3
```

Note that the inner loop is completely executed once for each step of the outer loop.

Care must be taken to be sure the loops are properly nested and not overlapped as would occur if lines 40 and 50 above were reversed. Lines 40 and 50 can also be written in a shorthand form as " 40 NEXT J,I". If a FOR-NEXT loop is exited with a GOTO statement and another FOR-NEXT loop is entered using the same variables as the first loop, then BASIC discards all FOR-NEXT loops that have executed between the first and second loops. When attempting to return to the first loop, an error will occur.

Exiting in the middle of FOR-NEXT loops and then reusing the same loop variables as loop variables can create unexpected NEXT without FOR errors. Such errors can be avoided by using different loop variables.

STOP Statement

Program execution is halted with a STOP statement. Its form is:

```
N STOP
```

where N is a line number. There may be more than one STOP statement in a program. When execution is halted a BREAK message with the line number is printed. For example,

```
10 PRINT "HERE"
20 STOP
```

results in the output:

```
HERE BREAK IN 20
```

A program that has been halted by a STOP statement can be restarted where it left off by the CONT command.

END Statement

The END statement is often used as the last statement in a program. Like the STOP statement, it terminates execution. It has the form:

```
N END
```

where N is a line number. The END statement is optional. If used, it need not be the last statement in the program, and there may be more than one END statement. In contrast to the STOP statement, no BREAK message is printed and the program cannot be restarted where it left off.

PROGRAMMING

BASIC Programs

A BASIC program can be entered into the computer whenever the BASIC prompt

```
    OK
or
    ok
```

appears on the screen. BASIC program is composed of lines. Each line begins with a number (called the line number) followed by a list of BASIC statements separated by colons. For example,

```
10 A = 60
20 B = A/2 : C = A/3
30 PRINT B, C
```

Any integer from 1 to 63999 can be used as a line number.

The lines of the program are typed one at a time. A line can hold 71 characters. A statement is not allowed to overlap two lines.

After each line is typed, the <RETURN> key is depressed.

After all of the program has been entered, it is executed by typing the command

```
RUN
```

without a line number and followed by <RETURN>. Normally, the statements in a program are executed starting with the lowest line number and then to the next lowest, and so on. Statements on the same line are executed in order from left to right.

If the program listed above were run, the screen would appear as follows:

```
OK
10 A = 60
20 B = A/2 : C = A/3
30 PRINT B, C
RUN
 30                    20

OK
```

The program is entered after the OK prompt. The RUN command is given and execution proceeds as follows:

Line 10: The value of 60 is assigned to the variable A.

Line 20: The value of A is divided by 2 and assigned to the variable B (now 30). Then the value of A is divided by 3 and assigned to the variable C (now 20).

Line 30: The values of B and C are printed.

The output consists of the two numbers 30 and 20. The OK prompt appears after execution is completed. The computer is ready for another instruction.

The program is stored in a region of memory referred to as the workspace. The program will remain in the workspace until erased by the NEW command, replaced by a program loaded from tape or disk, or lost by rebooting or unplugging the computer.

Immediate Mode

As each line is typed it is stored in a memory location referred to as a buffer. If the line begins with a line number it is added to the program in the workspace. If the line does not begin with a number it is executed immediately. This immediate execution feature is called the immediate mode or calculator mode of BASIC. Most BASIC statements can be used in the immediate mode. Examples:

```
PRINT SIN(.315)
```

and

```
FOR I=1TO100:PRINT 1 3:NEXT
```

Commands A command is an instruction that is usually used in the immediate mode, as opposed to a statement which is an instruction that usually appears within a program.

NEW Command: If a program resides in the workspace, any new lines that are entered with a line number will be added to it. In order to create new programs, the workspace must be cleared by the NEW command. It has the form

```
NEW
```

Because the NEW command returns from a BASIC program to the immediate mode, it is not as useful as other commands when used within a program. But the NEW command can be used within a program for read protection by using NEW in place of an END command.

RUN Command: The RUN command has the following forms:

```
RUN
RUN M
RUN"N
```

where M is a line number and N is the name of a program stored on disk. The first form starts execution of the program in the workspace at the lowest line number. The second form starts execution of the program in the workspace at the line numbered M. The third form causes the program named N to be loaded from disk and executed starting at the lowest line number. Examples:

```
RUN 135
RUN"ACCOUNT
RUN"23
```

All forms of the RUN command can be used within a program. When used in a program, the first form simply restarts the program; the second form acts in a manner similar to the GOTO statement except that the variable table is cleared. As variables are encountered in a program, they are put into a variable table along with their values. The third form can be used to "chain" programs so that they are executed one after the other. Programs are chained by having a statement in each program be a RUN command giving the name of the next program to be executed.

LIST Command: The LIST command causes a segment of the program in the workspace to be printed, usually on the screen. It has the forms:

```
1) LIST
2) LIST F
3) LIST#D
4) LIST#D,F
```

The first form lists the entire program on the screen.

The letter F represents one of the following forms which are illustrated by example using the second form above:

LIST 10	lists only line 10
LIST -10	lists from the beginning to line 10
LIST 10-	lists from line 10 to the end
LIST 10-20	lists from line 10 to line 20

The letter D is an output device number. If a hardcopy printer were device number 5, then the entire program would be printed by

```
LIST#5
```

and lines 10 through 20 would be printed by

```
LIST#5, 10-20
```

Device numbers are discussed in this chapter under "Device Specified Input and Output".

It is often desirable to "page through" a program by stopping and restarting the LIST command. The listing can be halted by <CTRL/S> and restarted by <CTRL/Q>.

The output of LIST to the screen can be toggled in by <CTRL/O>. This differs from the features described above in that the listing continues; it simply does not appear on the screen.

Since LIST returns from a BASIC program to the immediate mode it is not very useful as a program statement.

CONT Command: The continue command has the form:

```
CONT
```

The continue command can be used only in the immediate mode.

When a program is halted by a <CTRL/C> or STOP statement, a pointer is set in the program at the point of interruption. The program can be restarted where it left of by the CONT command. Restarting the program need not take place immediately. For example, the immediate mode can be used for LISTing and PRINTing without disturbing the pointer. This provides the programmer with a very useful debugging procedure:

1) Place STOP statements at convenient points within the program.

2) RUN the program.

3) When a STOP is executed a BREAK message with the line number will be printed. The program segment that was just executed can be listed using the LIST command, and the present values of variables can be determined by using the PRINT statement in the immediate mode. For example.

```
LIST 100-200
PRINT A,B,C$
```

NOTE: If a new line of text is added to the program, the continuation pointer is cleared and a CN (Continue) error will be given if CONT is used.

Interrupting Execution

A program that is running can be halted by depressing <CTRL/C>. It can be restarted where it left off with the CONT command.

The keyboard is continually checked during execution to see if a <CTRL/C> has been depressed. This feature can be disabled by a FLAG 25 command.

The LIST command can also be halted; see LIST COMMAND above.

Editing a Program

Corrections can be made in a line as it is being typed. An underscore, delete or backspace will backspace and delete the last character. Multiple deletions can be made by repeating the <SHIFT/O>.

A line can be deleted as it is being typed by entering a "commercial at" symbol, @.

If a new line is entered with the same line number as a previous line, it will replace the previous version. Thus a line can be removed from a program by simply typing its line number followed by <RETURN>. Program line editing is discussed more in-depth in Chapter Six.

REM Statement

Remarks can be placed in BASIC programs with the REM statement. These comments can often be very useful to a person reading the program. They are ignored by the computer when the program is executed.

The remark statement has the form:

```
N REM R
```

where N is a line number and R is a remark. For example:

```
100REM SUBROUTINE TO FIND RATIO
250 X=T/D:REM X IS THE RATIO
```

As shown by the above examples, a remark may appear as the only statement on a line or follow other statements. However, another statement should not follow a REM statement on the same line. It would not be executed; everything after REM is ignored on execution.

Saving Space

Occasionally, program additions cause a BASIC program to increase in size to the point that it will not fit into the available workspace. BASIC provides some features that can be of help.

FRE Function: The amount of workspace available to the programmer can be determined by the free function. The free function returns the number of bytes of memory in the workspace that are unused. It has the form:

```
FRE(X)
```

where X is a dummy variable. A programmer who wishes to expand an existing program should run the program before using the free function; additional memory is required during execution for the variable table. After the program is executed, the following line can be entered in the immediate mode:

```
PRINT FRE(X)
```

If more than 32K of memory is available, the FRE function returns a value that has cycled negative. That is, values increase in the order 1,2,...,32767,-32768,....When FRE(X) is negative, the number of available bytes can be determined by

```
PRINT 65536+FRE(X)
```

CLEAR Statement: As variables are encountered in a program they are put in a variable table along with their values. The clear statement clears the variable table and RESTOREs the DATA pointer. It has the form:

```
N CLEAR
```

where N is a line number.

The following example illustrates the effect of the CLEAR statement. The FRE function is used to determine the amount of workspace remaining unused.

```
 10 PRINT FRE(X)
 20 A=2:A$="X"
 30 PRINT FRE(X)
 40 PRINT A,A
 50 CLEAR
 60 PRINT FRE(X)
 70 PRINT A,A
OK
RUN
 24496
 24482
 2            X
 24496
 0
```

Another means of clearing variables is with the KILL command. It is discussed in Chapter Six of this manual.

Suggestions: The first place a programmer can look for additional space is the overall design of the program. After that, some simple fixes can be tried. For example:

1) Use subroutines for repeated code and functions for repeated calculations.

2) Remove blanks:
```
10 FOR I = 1 TO 10
```
and
```
10 FORI=TO10
```
are equivalent.

3) Remove REM statements.

4) Remove line numbers by putting more statements per line.

5) Reuse variable names.

6) Use smaller names such as A for A1.

7) Put variables in arrays; an array of 10 elements uses less space than 10 different variable names.

8) Integer arrays use less space than 10 different real arrays.

ARRAYS

Large quantities of data can be handled in BASIC by organizing the data into arrays. Arrays can be one -- or multi-dimensional. The elements of an array are subscripted variables.

Subscripted Variables

Variables may be simple or subscripted; up to 255 subscripts are allowed in an array. Subscripted variable references have the form

N(L)

where N is an arithmetic, integer, or string variable name; and L is a list of arithmetic expressions, called subscripts, separated by commas. Examples:

```
N1(2)
V$(5.6,4*X)
RA%(S,T,W)
```

The arithmetic expressions used as subscripts are evaluated and then truncated to integer values. Subscripts can have values between 0 and 32767, inclusive. If the array is not DIMensioned before it is used, the default subscripts are 0-10. Larger subscript values are allowed if the array is dimensioned in a DIM statement.

DIM Statement

The dimension statement has the form

N DIM L

where N is a line number and L is a list of subscripted variable names. For example,

```
10 DIM A(20),B$(1,2)
20 DIM X1(N*2)
```

The array A is a one-dimensional arithmetic array having twenty-one elements: A(0),...,A(20). The array B is a two-dimensional string array having the six elements: B$(0,0), B$(0,1), B$(0,2), B$(1,0), B$(1,1), and B$(1,2).

Arrays can have variables as subscripts. For example:

```
10 INPUT"WHAT IS THE DIMENSION OF M";N
20 DIM M(N)
```

Dimension statements are usually placed together at the beginning of the program. Subscripted and simple variables can be DIMensioned. This is a good practice when using a few often-used variables out of a lot of possible ones. It will put them at the front of the variable space and search through them first. However, dimension statements can occur anywhere in a program. Space is allocated as they are encountered. The dimension statement must be executed before the array is used. A double dimension error occurs if an array is encountered in a DIM statement after one of the elements of the array has been encountered. A double dimension error also occurs if an array is encountered in more than one DIM statement.

FUNCTIONS

Mathematical Functions

The following mathematical functions are supplied in OS-65U BASIC. In general, the arguments of these functions may be any arithmetic expression. Exceptions are noted in the discussion of each function.

ABS Function: The absolute value function returns the absolute value of its argument: ABS(X) is equal to X if X is greater than or equal to zero and ABS(X) is equal to -X if X is less than zero. For example, ABS(-7.5) is 7.5.

EXP Function: The exponential function returns e=2.71828...raised to the power of its argument. For example, EXP(1) is e. The argument of EXP must be less than 88.0296919.

INT Function: The integer function returns the greatest integer less than or equal to its argument. For example, INT(6.6) is 6 and INT(-3.2) i -4.

LOG Function: The logarithm function returns the natural logarithm (log to the base e) of its argument. The argument must be positive. The log to another base, say B, of X is LOG(X)/LOG(B).

RND Function: The random number generating function returns a number between 0 and 1. This function is usually used to generate a sequence of pseudo-random values. For example, in ROM BASIC this program:

```
 5 X = RND(-1)
10 FOR I=1 TO 5
20 PRINT RND(1);
30 NEXT
```

results in the output:

```
.163997         .56961          .865247         .323602         .412642
```

If the argument is positive, it is a dummy argument. That is, its value is not important; RND only checks to see if it is positive. As long as the argument remains positive, RND will generate the next number in the sequence using the last value returned. The random number sequences are periodic. The example above repeats after 1861 calls to RND.

If the argument is negative, RND will start a new sequence with a new period based on the value of the argument. Thus negative arguments serve as seeds. The same sequence is generated if the same seed is used.

If the argument is zero, RND will return the previous value again.

If the programmer wishes to have a program generate a different random number sequence each time the program is run, he should devise a procedure for choosing the seeds. Such a procedure might be based on PEEKing various memory locations.

A random number N between two numbers A and B (A<N<B) can be obtained by N = A=RND(X)*(B-A).

SGN Function: The sign function returns the sign of the argument. Plus one is returned for positive arguments, minus one for negative arguments, and zero is returned if the argument is zero.

SQR Function: The square root function returns the square root of its argument. For example, SQR(4) is 2. The argument must be positive.

Trigonometric Functions: The sine, cosine, tangent, and arctangent functions are supplied by BASIC. They are called by the following forms: SIN(X), COS(X), TAN(X), AND ATN(X); where the argument is an arithmetic expression. The value of the argument of ATN(X) must be between -1 and 1.

The trigonometric functions require their arguments to be in radians. To convert degrees to radians: radians

```
SEC(X) = 1/COS(X)

CSC(X) = 1/SIN(X)

COT(X) = 1/TAN(X)

ARCSIN(X) = ATN(X/SQR( -X*X+1))

ARCCOS(X) = ATN(X/SQR( -X*X+1)) +1.5708

ARCSEC(X) = ATN(SQR(X*X-1))

ARCCSC(X) = ATN(1/SQR(X*X-1)) +(SGN(X)-1)*1.5708

ARCCOT(X) = ATN(1/X)
```

```
SINH(X) = (EXP(X)-EXP(-X))/2

COSH(X) = (EXP(X)+EXP(-X))/2

TANH(X) = EXP(-X)/(EXP(X)+EXP (-X)) *(-2)+1

SECH(X) = 2/(EXP(X)+EXP(-X))

CSCH(X) = 2/(EXP(X)-EXP(-X))

COTH(X) = EXP(-X)/(EXP(X)-EXP (-X))*2+1

ARCSINH(X) = LOG(X+SQR(X*X+1))

ARCCOSH(X) = LOG(X+SQR(X*X-1))

ARCTANH(X) = LOG((1+X)/(1-X))/2

ARCSECH(X) = LOG((SQR(-X*X+1) +1)/X)

ARCCSCH(X) = LOG((SGN(X)*SQR(X*X+ 1)+1/)X

ARCCOTH(X) = LOG((X+1)/(X-1))/2
```

String Functions

A string function is either a function whose argument is a string or a function which returns a string. String functions may return either numeric values or strings. Those that return strings have names ending with a dollar sign.

The following functions, while not intrinsic to BASIC, can be calculated using the existing BASIC functions as follows:

ASC Function: The ASCII function returns the ASCII value in decimal of the first character in the argument. It has the form

```
ASC(X$)
```

where X$ is a string expression. For example, ASC("BIG") is 66.

CHR$ Function: The character function returns a one-character string. The character returned is the one whose decimal ASCII value is the argument. It has the form

```
CHR$(X)
```

where X is an arithmetic expression whose value is between 0 and 255. The character function is essentially the opposite of the ASC function. For example, CHR$(66) IS "B".

LEFT$ Function: The left function returns a left-most substring of a string. It has the form

```
LEFT$(X$,Y)
```

where X$ is a string expression and Y is a positive arithmetic expression. The Y left-most characters of X$ are returned. For example, LEFT$("123456",3) is "123". If Y exceeds the length of the string, the whole string is returned.

LEN Function: The length function returns the length of a string. It has the form

LEN(X$)

where X$ is a string expression. For example, LEN("OUT") is 3.

MID Function: The middle function returns a middle substring of a string. It has the two forms

MID$(X$,Y)
MID$(X$,Y,Z)

where X$ is a string expression, Y is a positive arithmetic expression and Z is a nonnegative arithmetic expression. The first form returns the substring of X$ starting in the Y-th position and continuing to the end of the string. For example, MID$("123456",3) is "3456". The second form returns a substring of length Z starting in the Y-th position and continuing for Z characters. For example, MID$("12345",3,2) is "34". If Y exceeds the length of the string, the string of length zero, "", is returned. If Z goes past the end of the string, the substring starting in the Y-th position to the end of the string is returned.

RIGHT$ Function: The right function returns a right-most substring of a string. It has the form

RIGHT$(X$,Y)

where X$ is a string expression and Y is a positive arithmetic expression. The right-most Y characters of X$ are returned. For example, RIGHT$("VALUE",3) is "LUE". If Y exceeds the length of the string the entire string is returned.

STR$ Function: The string function returns the argument as a string. It has the form

STR$(X)

where X is an arithmetic expression. For example, STR$(12.3) is " 12.3", a string of length 5. For positive numbers, a leading blank instead of a plus sign is returned. The results are the same as when X is PRINTed; except that no trailing blank is included in the string. Some forms are converted; for example,

```
10 PRINT"12345678901234567890"
20 A(1)=15.1
30 A(2)=-25
40 (3)=12.0E+2
50 A(4)=10000000000
60 FOR I = 1 TO 4
70 A$=STR$(A(I))
80 PRINT A$, LEN(A$) : NEXT
```

results in the output

```
12345678901234567890
 15.1          5
-25            3
 1200           5

1E+11     6
```

VAL Function: The value function returns the numeric value of a string. It is the opposite of the STR$ function. Its form is

```
VAL(X$)
```

where X$ is a string expression representing a number. For example, VAL("0.0035") is 3.5E-03. If X$ does not represent a number the value 0 is returned.

SUBPROGRAMS

A calculation that needs to appear more than once in a program can be written as a subprogram. The subprogram can be called each time it is needed; thus avoiding the necessity of rewriting the calculation. There are two types of subprograms in BASIC: statement functions and subroutines. A statement function consists of a one-statement calculation. A subroutine can be a self-contained program.

Statement Functions

In addition to the functions supplied by the system, the user can create functions called statement functions. Statement functions are defined by the DEF statement.

DEF Statement: The define function statement has the form

```
N DEF FNX(A) = E
```

where N is a line number, X and A are simple numeric variables, and E is an arithmetic expression. The name of the function consists of the letters FN followed by a variable name. The variable A is called the dummy variable. The expression E may reference other functions; including those defined by DEF statements. The use of the variables X and A in the function does not affect their use elsewhere in the program. The define function statement cannot be used in the immediate mode.

Consider the following program:

```
10 DEF FNS(P) = P + P  2
20 X = 2
30 PRINT FNS(X+1)
```

The output is

```
12
```

The function FNS is defined in line 10 to be P plus the square of P. In line 30, the programmer has replaced the dummy argument P by an arithmetic expression, X + 1. At this point in the program the value of X + 1 is 3, so 3 is substituted for each occurrence of P in the defining expression. The result, 3 3 2 = 12, is assigned to FNS(X+1).

Subroutines A subroutine consists of a program segment ending with a RETURN statement. A GOSUB statement calls the subroutine by transferring control to the first line of the subroutine. When the return statement at the end of the subroutine is encountered, control returns to the statement following the GOSUB statement.

GOSUB and RETURN Statements: The form of the GOSUB statement is

```
N GOSUB M
```

where N and M are line numbers. The directive M can also be a variable. Control is transferred to line number M. The form of the RETURN statement is

```
N RETURN
```

where N is a line number. For example,

```
10 PRINT "START"
20 GOSUB 50
30 PRINT "OUT OF SUBROUTINE"
40 END
50 PRINT "IN SUBROUTINE"
60 RETURN
```

results in the output

```
START
IN SUBROUTINE
OUT OF SUBROUTINE
```

Subroutines can call other subroutines, including themselves. Subroutines may have logical branches each of which ends in a RETURN statement.

It is convenient to picture the transfer of control as follows: As the GOSUBs are encountered they are stacked one on the other; when a RETURN statement is encountered one of the GOSUBs is peeled off the top of the stack. Control then passes to the next statement following the GOSUB that was on top of the stack. In general, subroutines placed at the beginning of a program will be found more quickly than those at the end. BASIC searches for subroutines by asking if the GOSUB line is greater or less than the current line. If it is a greater number, BASIC searches the rest of the program for the line. If the line is a lesser number, BASIC goes to the beginning of the program to search for it.

If a RETURN statement is encountered with no GOSUB on the stack, a RETURN-without-GOSUB error occurs. For this reason, subroutines are placed after a STOP or END statement denoting the end of the main logical sequence in the program.

ON...GOSUB STATEMENT: The ON...GOSUB statement is a conditional transfer statement similar to the ON...GOTO statement. It has the form

```
N ON E GOSUB L
```

where N is a line number, E is an arithmetic expression, and L is a list of line numbers separated by commas. Each line number can also be represented by a variable. When an ON...GOSUB is processed, the expression E is evaluated and truncated. Control then passes to the E-th line number in the list L. When a RETURN is encountered, control returns to the statement following the ON...GOSUB statement. For example,

```
20 ON Z GOSUB 100,200,300
```

transfers control to statement 100 if Z = 1, to statement 200 if Z = 2, and to statement 300 if Z = 3. If Z is less than 1 or greater than 3 then the ON...GOSUB statement is ignored and control passes to the following statement.

DIRECT MEMORY CONTROL

The following features of BASIC can be very helpful to the experienced programmer. Care must be exercised with these statements and functions because they access the memory of the computer directly. An improper operation with any of these commands can cause a system crash, wiping out BASIC and the user's programs.

CAUTION

PEEKs and POKEs that alter the operating system other than those described in this manual should be used with extreme discretion. M/A-COM OSI cannot guarantee that any future version of OS-65U will be compatible with PEEKs and POKEs not documented in this manual.

POKE Statement

The POKE statement stores a value into a memory location. It has the form

```
N POKE, I,J
```

where N is a line number, and I and J are arithmetic expressions whose values are integers. The value of I is a memory location expressed in decimal and the value of J is placed in location I. The value of J must be between 0 and 255 inclusive. For example,

```
10 FOR I = 14822 TO 14828
20 POKE I, A(I)
30 NEXT
```

stores the array A in the memory locations 14822 to 14828.

The expression J cannot contain the PEEK function. No error results; the value is not POKED.

PEEK Function

The PEEK function reads a memory location. It functions as the opposite of the POKE statement. It has the form

```
PEEK(I)
```

where I is a memory location or I/O location expressed in decimal. For example,

```
10 FOR I = 14822 TO 14828
20 A(I) = PEEK(I)
30 NEXT
```

assigns the values in memory locations 14822 to 14828 to the array A. A memory location can store one byte; the value returned by the PEEK function is therefore an integer between 0 and 255.

PEEKing memory locations where hardware or memory locations do not exist will return unpredictable values.

WAIT Function

The WAIT function halts program execution or causes the program to "wait" until a particular bit in memory is set or reset. It has the forms

```
WAIT I,J
WAIT I,J,K
```

where I is a memory location expressed in decimal, and J and K are integers between 0 and 255. The WAIT function in the first form reads the status of memory location I, then ANDs the result (see "The Bit Operators - AND, OR, NOT") with J until a nonzero result is obtained.

The WAIT function in the second form reads the status of memory location I, exclusive ORs that value with K, and then ANDs that result with J until a nonzero result is obtained. The bit operator OR compares two binary numbers bit by bit and sets a bit in the result to 1 if one or both of the corresponding bits in the two numbers are 1. The logical operation "exclusive" OR is similar, but sets a bit to 1 in the result if exactly one of the corresponding bits in the two numbers is 1.

The WAIT function is used for fast service of input status flags. For example,

```
WAIT X, 1
```

will cause the execution of a BASIC program to halt and then continue when bit zero of memory location X goes low.

DISK INPUT AND OUTPUT

OS-65U contains a complete disk operating system. Below is a brief description of the data and program file commands available in the system.

Data and Program File Commands

OPEN Command: The OPEN command is used to OPEN a data file for access. The command has two possible formats:

```
OPEN "filename","password",channel number
     and
OPEN "filename",channel number
```

The channel number must be an integer between one and eight. After OPENing a file, the file is referred to by the channel it was opened under rather than by its name. If a file is opened with the correct password, the system may read or write to the file regardless of that file's access rights. If a file is opened without the correct password, the system may access the file only in the manner defined by its access rights (e.g. read only, write only, none, etc.). Once a file has been OPENed under a channel, it is not necessary to specify the DEVice name to access the file; the channel name is enough.

CLOSE Command: The CLOSE command causes a file to be closed. Closing a file frees the channel the file was opened under. Simply typing

```
CLOSE
```

closes all the channels currently open. Typing

```
CLOSE X
```

where X is a specific channel number, closes only that channel.

INDEX Command: The INDEX is a reserved system variable. This command takes two forms; the first is

```
INDEX (channel number)
```

An example of the program usage is

```
X=INDEX(1)
```

After the Execution of this statement, X equals the value of the INDEX for channel 1. This INDEX value is simply a relative pointer into the file opened under a particular channel. When a file is opened, the INDEX of the channel the file was opened under is set to zero.

The second form of the INDEX function is

```
INDEX <channel number> = expression
```

This form permits "bumping" the INDEX to point anywhere within the data file. The INDEX function permits the user to define the file formats to be whatever is desired. See Chapter Four of this manual for a more complete description of the use of the INDEX statement.

PRINT% Command: This command has the form:

```
PRINT% channel number, variable expression
```

where the variable expression may be string or numeric. This statement directs the variable to be printed into the data file opened under the channel "channel number". The variable(s) will be printed into the file starting at the current position of the INDEX of that channel.

INPUT% Command: This command has the form:

```
INPUT   channel number, variable(s)
```

This command INPUTS data from the file opened under "channel number" and assigns the data to the variable(s) that appear in the INPUT% statement. The data is INPUT from the file starting at the current position of the INDEX. The INPUT% statement of a variable terminates upon the receipt of a the variable.

FIND Command: The FIND command has the form

```
FIND "string expression," channel number
```

The FIND command executes a high speed search for the string expression in the file opened under "channel number". The search starts from the current position of the INDEX for that channel. If the string is found, the INDEX for that channel points at the string in the file. If the string is not found, the INDEX for that channel is set to 1E9 (1,000,000,000). Note that the FIND command may be used to find subsets, for example,

```
FIND "ABCD",1
```

would find the subset "ABCD" in the string "ABCDEFG". The FIND command also permits "don't care" characters within the string it is searching for. That is,

```
FIND "AB&HI",1
```

would find "ABCHI" and ABZHI".

A literal ampersand may not be used.

FLAG Command: The FLAG command permits the user to taialor his system toward a specific application. The FLAG command has the form:

```
FLAG N
```

where N is the flag number of the option desired. The options and their flag numbers are presented in Chapter Five of this guide.

DEV Command: The device command specifies which device is to be the current mass storage unit. It has the form:

```
DEV"U$"
```

where U$ is the unit. Unit specifications are A, B, C or D for floppy disks, E or F for hard disks, and K through Z for machines connected in a network. OS-65U DEVices are explained fully in Chapter Two of this manual.

LINKING PROGRAM TO MACHINE LANGUAGE ROUTINES

The USR function permits leaving a BASIC program, executing a machine language routine, and then returning to the BASIC program.

USR Function

The USR function has the form

```
USR(X)
```

where X is an arithmetic expression. The value of X can be sent to the machine language routine, and a single value can be returned as USR(X).

If no parameters are passed, the function is used in the form

```
N Y = USR(X)
```

where N is a line number, and X and Y are dummy variables. Control passes to the machine language routine at line N and then returns to the next line. It is often more convenient to use the second form and pass parameters by PEEK and POKE rather than to use the parameter passing feature of the USR function. If no parameters are passed, Y is assigned the value of X.

Before the machine language routine can be called by the USR function, its starting address must first be POKEd into memory.

```
POKE 8788, LO and
 POKE 8779, HI
```

For example, if the routine starts at $6000 then 60 is the high byte and 00 is the low byte. Converting to decimal, HI is 96 and LO is 0.

Passing Parameters: The machine language routine begins by calling a routine whose starting address is a $0006. This routine converts the argument X into a 16-bit two's complement number which is then stored. The storage location of this number is as follows:

HIGH BYTE	LOW BYTE
$00B1	$00B2

The value of X is now available for the machine language routine.

The machine language routine ends by placing the value to be returned to the BASIC program in the accumulator (high byte) and the Y register (low byte); it then calls a subroutine that starts at $0008. This subroutine will pass the value to the BASIC program as USR(X) and then return control to the BASIC program.

An example of the USR Function is given in Chapter Seven of this Guide.

FILE ORGANIZATION AND PROCEDURES

Keyware OS-65U users are provided with a good deal of flexibility in structuring data files. This chapter describes the basic structure of OS-65U files and some of the more useful data file structures available to the programmer; advanced tools available for accessing and enhancing the use of the data files is also described. An example of OS-65U advanced file structures and programming techniques is provided by a description of the files used in the M/A-COM OSI database management system, Keybase.

The file structures described in this chapter are called Sequential, Direct Record and Direct Field. The fundamental Keyware OS-65U file organization is a standard sequential format consisting of variable-length fields separated by carriage returns. File structures discussed in this chapter that can be programmed from this sequential format are defined as follows:

- Direct Record. This structure uses fixed-length records with variable-length fields. It provides direct access to records and sequential access to fields in the records.
- Direct Field. This structure uses fixed-length records and fixed-length fields. It provides direct access to both records and fields.

It should be noted that the file structures discussed in this chapter have proven to be highly effective, but the programmer is not limited to them. Keyware OS-65U provides a good deal of flexibility for almost any file structure dictated by the application.

DATA CHANNELS

It is necessary, when accessing a file, to associate a channel number (1-8) with it. This channel is analogous to a path that is used to reach the file. Figure 4-1 illustrates a file as the hub of eight channels, any one of which can be used to reach it.

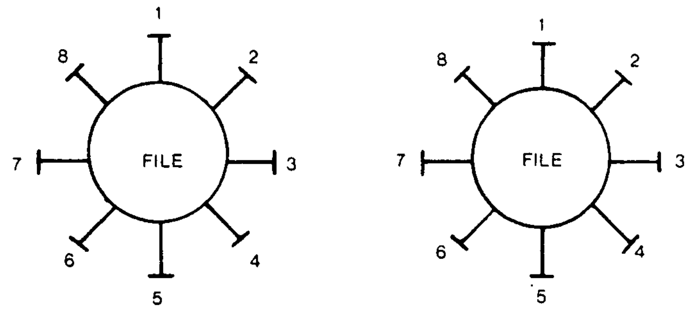

FIGURE 4-1

A file is associated with a channel by OPENing it on that channel. Only one file can be associated with a given channel at any one time. OPENing a file on a channel performs various necessary functions; it creates a record in memory of the file's starting address and length, retains the file's access rights, remembers the device the file is on, etc. After a file has been OPENed on a channel, giving the channel number is sufficient to access that file.

A file is disassociated from a channel by CLOSEing the file. CLOSEing a file records any changes made to the file and terminates its association with the channel. The channel is then available for use with the same or another file.

BASIC FILE COMMANDS

BASIC commands used in entering and retrieving file information are listed below by verb, syntax and brief definition. More complete definitions of the commands are provided in this chapter and in Chapter Three.

OPEN	OPEN "FNAME","PASS",CH OPEN "NAE",1 OPEN FN$,PW$,CH	Opens a data file on a given channel.
CLOSE	CLOSE CH CLOSE	Closes an open file. If no channel number specified, closes all open files.
INPUT	INPUT%CH, F$	Causes BASIC to read data from a disk file.
PRINT	PRINT%CH,"DISC OUTPUT"	Causes BASIC to write data a disk file.
INDEX	INDEX<CH>=N N=INDEX(CH)	Sets or determines an open disk channel's file index position. INDEX<CH>=N sets current index position, N=INDEX(CH) returns to current index position.
FIND	FIND "LOAN",CH FIND A$, CH	High speed search in the disk file for the string expression on the specified channel number. Channel number is set by INDEX <CH>.

FILE MANAGEMENT PROCEDURES

Techniques for OPENing, CLOSEing files and performing end-of-file tests are the same for various types of OS-65U files, whether Sequential, Direct Record or Direct Field. They are discussed below.

Opening a File

The syntax for opening a Keyware OS-65U data file is

```
OPEN F$,CH
OPEN F$,P$,CH
```

where F$ is the file name, CH is the channel number and P$ is the password. Recall that files may be created with a choice of access rights which include the choices of Read only, Write only, Read/Write and None (no access). If a file that has a password is OPENed with the wrong password (or no password), the access rights take effect. If the file is OPENed with the proper password, the access constraints do not apply. For instance, if a file were created with "Read Only" access, writing to the file would be allowed only if the file were OPENed with the correct password.

Closing a File

The syntax for closing a Keyware OS-65U data file is

```
CLOSE CH
CLOSE
```

where CH is the channel number. If no channel number is specified, all open channels are closed. A CLOSE command that specifies a channel that has not been opened will produce an error.

End-of-File Testing

Testing for the end of the file is accomplished automatically by the operating system. The next section on file INDEXes discusses how the INDEX automatically increments to the next location in the file and can be used as a pointer to where the next action in the file is to occur. Any attempt to INPUT or PRINT past the physical end of a file results in an end-of-file error. This automatic end-of-file test can be overridden by the use of FLAG 5; see the chapter on FLAGs for a description of its use. Note, however, that if that FLAG 5 is used, it becomes the responsibility of the program to test for the end of the file (1E9).

THE FILE INDEX

The INDEX is an integer that points to a location in the file. The file may be seen as a sequence of characters with every character in it sequentially numbered; the INDEX at the beginning of the string points to the numbered location desired. When that location is accessed, the INDEX automatically increments to the next location -- until terminating at a carriage return. If, when using an INPUT statement, a carriage return is encountered,the INPUT statement is terminated, leaving the INDEX pointing to the next location after the carriage return. In Figure 4-2, the INDEX points to zero when the file is opened. It increments at locations 1 through 4, terminates at the carriage return and points to the next location (5).

The INDEX statement requires the syntax

```
INDEX<CH> =X
X=INDEX(CH)
```

where the "INDEX( )" form is used to determine the current INDEX location and the "INDEX< >" form is used to set the INDEX.

The logical end of a file immediately follows the last location or record containing data -- as opposed to the physical end which may be well beyond the data in the file.

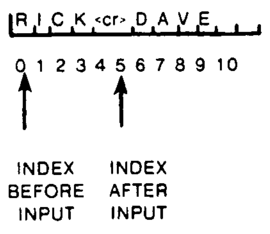

FIGURE 4-2

SEQUENTIAL FILES

Unless enhanced by procedures outlined in this chapter, Keyware OS-65U files are a straight sequential structure. A sample of this structure is illustrated by Figure 4-3. The data is arranged without records in an unbroken sequential string in variable-length fields. Since this type of file is difficult to update, it has a limited usage for most applications. It does, however provide a good basis from which to create more useful types of files.

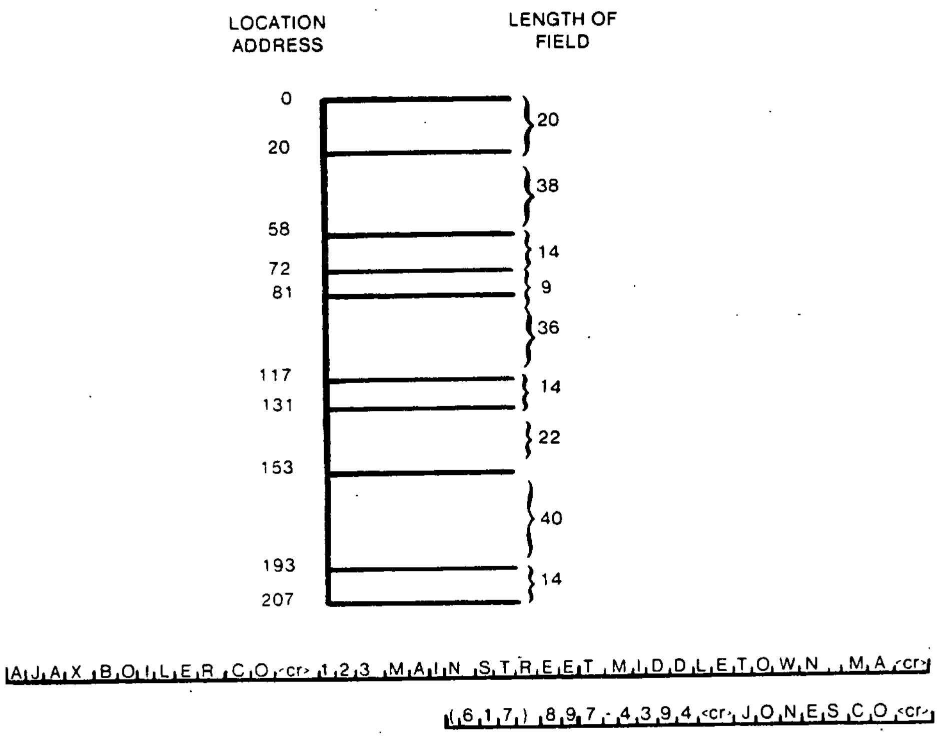
LOCATION ADDRESS
LENGTH OF FIELD
0
20
58
72
81
117
131
153
193
207
20
38
14
9
36
14
22
40
14
AJAX BOILER CO <cr> 123 MAIN STREET MIDDLETOWN MA <cr>
(617) 897-4394 <cr> JONES CO <cr>

FIGURE 4-3

DIRECT RECORD FILES

The Direct Record file structure, illustrated by Figure 4-4, uses fields of variable lengths within records of equal length. OS-65U does not use GET/PUT RECORD commands because the length of records is not preset and therefore is not known by the operating system; record lengths are established by the programmer. Records are accessed directly by the program using an INDEX as described in the next section.

The major drawback to the Direct Record structure is that it can be troublesome to make changes to a record. This is because the field lengths are variable and their starting locations can be ascertained only if the entire record is read. Read/Write operations on partial records within the file, therefore, involves a large overhead.

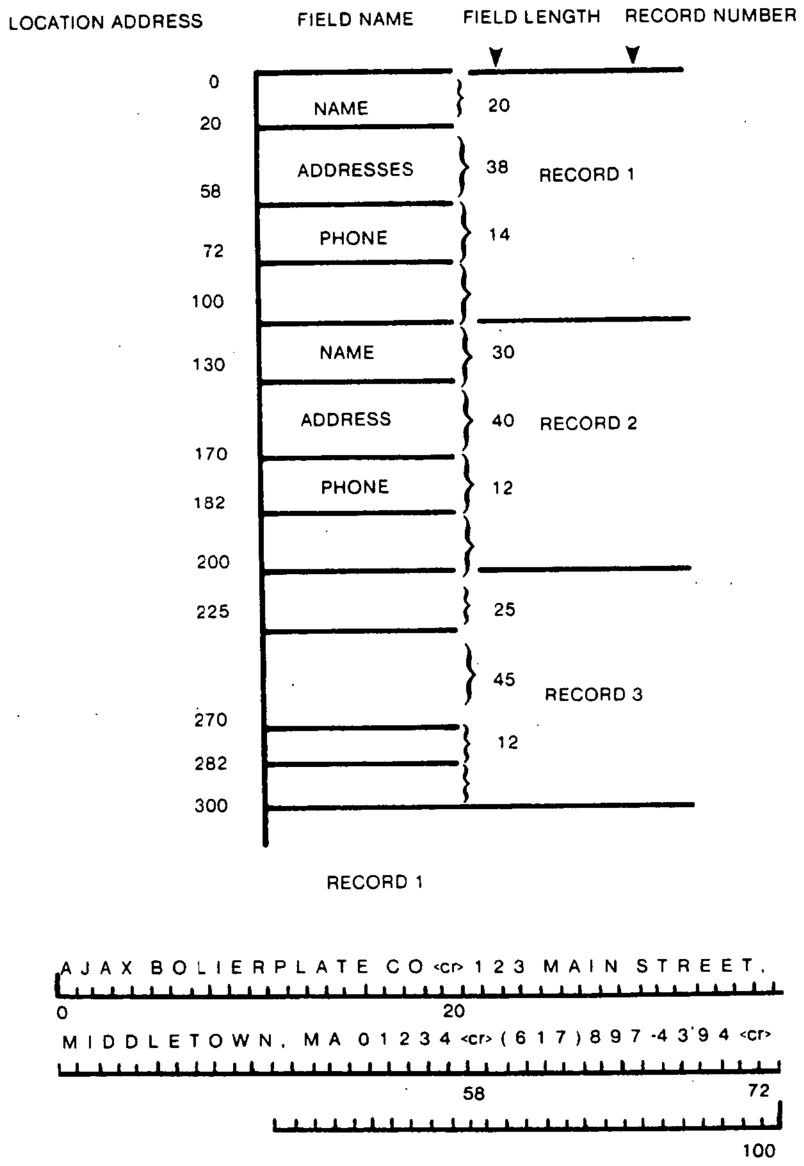

FIGURE 4-4

Using an INDEX With Direct Record Files

Using an INDEX statement to access Direct Record files is accomplished with the following syntax:

```
INDEX<CH>=(RC-1)*RL
```

where CH is the channel number, RC is the logical record number and RL is the length of the records. Sample programs are given below for reading a record in an Direct Record file and then writing to it. Assume the following variables have been set:

R	= 3
CH	= 1
RL	= 50
FN$	= File name
PW$	= Password
A$	= Field (Name)
B$	= Field (Phone)
C$	= Field (Address)

To access a phone number in record number 3, the following program would be required to read the file:

```
10 OPEN FN$,PW$,CH
20 INDEX<CH>=(R-1)*RL
30 INPUT%CH,A$,B$,C$
40 CLOSE
```

To write to the file and change the phone number:

```
100 OPEN FN$,PW$,CH
110 INDEX<CH>=(R-1)*RL
120 INPUT%CH,A$,B$,C$
130 B$=1-617-555-1212
140 INDEX<CH>=(R-1)*RL
150 PRINT%CH,A$
160 PRINT%CH,B$
170 PRINT%CH,C$
180 CLOSE
```

In the first program, line 20 sets the index to the start of the record and line 30 inputs both fields (since it is a sequential read). In the second program, line 130 sets B$ equal to the new phone number, line 150 prints the name, line 160 prints the new phone number and line 170 prints the address.

DIRECT FIELD FILES

OS-65U files may be organized into a direct access structure that retains a great deal of flexibility for changing records. This structure organizes the file into fixed-length records with fixed-length fields. Figure 4-5 illustrates this type of file. Note that the pattern of field lengths is the same through all the records in the file; in this example, the first field is always 35 bytes long, the second 49 bytes and the third 16 bytes. The structure of all records in the file is the same; making for fast INDEX calculations. Also, fields can be generally be modified without affecting other fields in the record. Note that record locations are relative to the start of the file while field locations are relative to the start of the record.

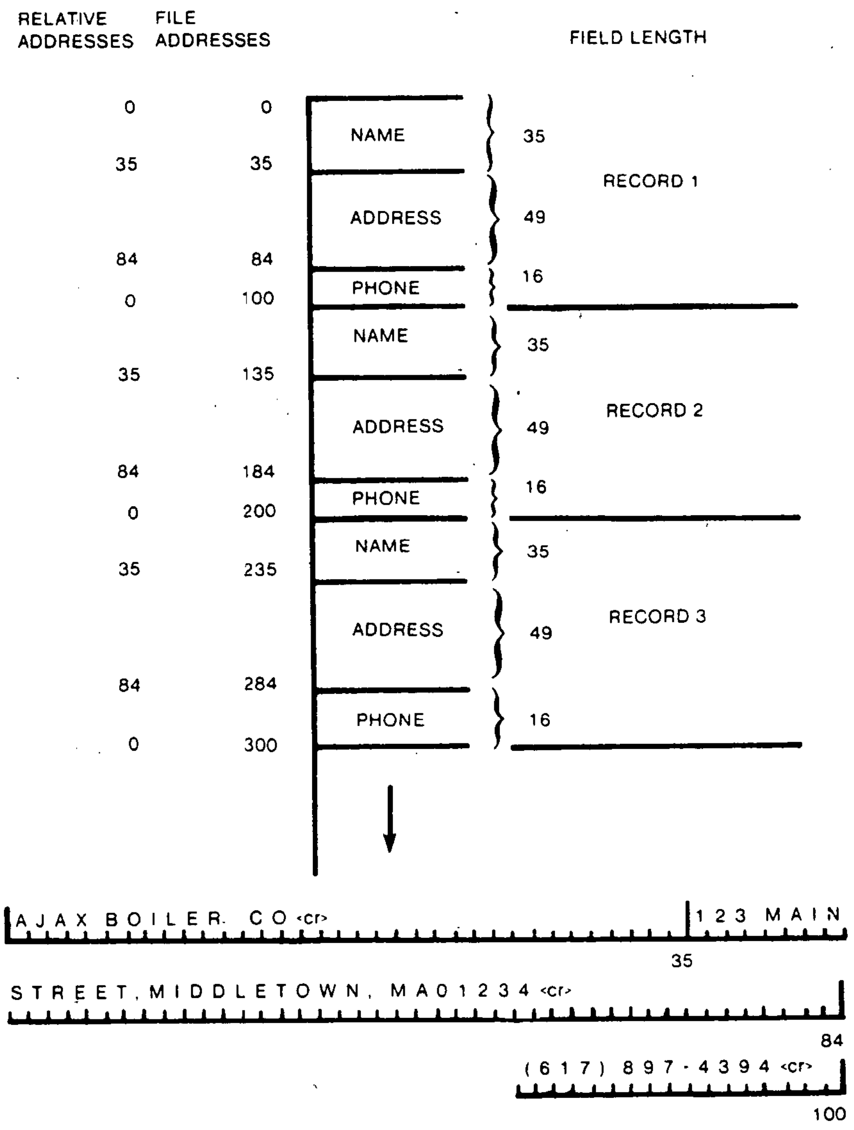

FIGURE 4-5

Using INDEX With Direct Field Files

Using the INDEX statement to access Direct Field files is accomplished with the following syntax:

```
INDEX<CH>=(R-1)*RL+FP
```

where CH is the channel number, R the logical record number, RL the record length and FP the field offset, that is, the relative location of the field. Sample programs are given below to read the file for a phone number and then write to the file to change it. In actuality, padding characters would have to be included in this formula (see next section); they have not been included here for the sake of simplicity. The example assumes the following variables have been set:

CH	=	1
R	=	3
RL	=	50
FP(1)	=	35
FP(2)	=	49
FP(3)	=	16
FN$	=	File name
PW$	=	Password

To read the file:

```
10  OPEN FN$,PW$,CH
20  INDEX<CH>=(R-1)*RL+FP(3)
30  INPUT%CH,B
40  CLOSE
```

B$ is now equal to the old phone number. To write to the file:

```
100  OPEN FN$,PW$,CH
110  INDEX<CH>=(R-1)*RL+FP(2)
120  B$="(617)555-1212"
130  PRINT%CH,B
140  CLOSE
```

Line 20 sets the Index at the start of the record and start of the field. Line 30 retrieves the old phone number. Line 120 sets B$ equal to the new phone number, and line 130 prints it out.

Using Filler Characters With Direct Field Files

Since some locations are unused in with Direct Field records, it is necessary that those locations in fields and records are filled in or "padded" in order to maintain the record and field addresses. As the file is written to, if all the locations are not padded, old data will remain in those locations, creating a lot of invalid data in the file. For instance, if a string occupying 50 locations were replaced by a string occupying 40 locations, the last 10 characters of the original string would remain until written over. Filling in or "padding" the field with some neutral character (such as a space) assures that the field is completely updated and the old data completely replaced. A good formula to set the padding mechanism is as follows:

```
10 IF LEN(SP$)<71 THEN SP$=SP$+" ":GOTO 10
```

This line instructs that if the length of the string is less than 71, then add a space, thus creating a string of spaces 71 characters long. 71 represents a good number to use as a filler, since OS-65U BASIC input has a maximum of 71 characters.

The second line of the program determines if the text is to be justified right (padding justified left) or the text justified left (padding justified right). This requires the syntax:

```
20 PRINT%CH,RIGHT$(SP$+B$,FL-1)
            or
20 PRINT%CH,LEFT$(B$+SP$,FL-1)
```

where %CH is the channel associated with the file, RIGHT or LEFT determines which direction the text is justified, B$ is equal to the data string being printed and FL-1 amounts to the field length minus one space for the carriage return. This instruction line concatenates the two strings, SP$ (the string of spaces) and B$ (the string of data) into one long string. FL-1 specifies the number of locations to take from (the left or right of) that concatenated string to insert in the field.

These string concatenations will process very slow, and create internal "garbage" in the system that will automatically be eliminated by BASIC; this causes a processing delay. This garbage collection can be avoided using an Extended PRINT command as explained in the Chapter Six of this guide.

THE FIND COMMAND

The FIND command initiates a high-speed sequential search to match a specified string in the file. Its syntax is

```
FIND S$,CH
```

where S$ is the string expression and CH the channel number. The string expression can be any string, e.g. an expression in quotes, a numeric value converted to a string, etc. The maximum length of string exnressions is 255 characters.

The search begins at the current INDEX for the channel, i.e., wherever the pointer is in the file, and continues until a match occurs or the physical end of the file is reached. If a match is found, the INDEX points to where it occurred. If no match occurs, the search will continue to the physical end of the file (searching empty records as well) and the INDEX will be >=1E9 (10 9 or 1,000,000,000). End-of-file tests must be included in a program that uses the FIND command. For example

```
IF INDEX(CH)>=1E9 THEN ...
```

The FIND command will attempt to match the INDEX value, or location, of the string specified; it can be an entire string or a subset of a string. If the FIND command is used, note that the fields in Direct Field files should be padded with characters (usually spaces) so the fields are entirely updated. Padding is discussed in more detail earlier in this chapter.

Enhancing the FIND Command

The capabilities of the FIND command can be enhanced by using the INDEX to discard all finds outside of a specified range of locations. As the FIND command searches, the program discards matches outside of the specified field (range of locations) in each record. For instance, in the file illustrated by Figure 4-5, if a particular string, such as a telephone area code, were being sought, then all occurences of the string found outside of the telephone field would not be of interest and therefore discarded. The following program provides a format whereby an INDEX is used to FIND all occurences of a specified string and retain only those found in a particular range:

Variables are defined as follows:

S$	=	The string being sought
BODF	=	Beginning of data field. See next section on Keybase files.
EODF	=	End of data file. See next section on Keybase files.
T	=	Temporary Variable
T2	=	Temporary Variable
T3	=	Temporary Variable
FP(2)	=	The range of locations covered by Field 2.
FL	=	Field Length

```
100 REM HERE WE SET THE INDEX BEFORE ATTEMPTING A FIND
110 :
120 INDEX<CH> = BODF
130 :
140 FIND S$,CH
150 IF INDEX (CH) >=EODF THEN PRINT "NOT FOUND": CLOSE: END
160 T = INDEX (CH) - BODF
170 T2 = INT ( T / RL ) * RL
180 T3 = T - T2
190 IF T3 < FP (2) OR T3> FP (2) +FL (2) -1 THEN GOTO 240
200 PRINT "FOUND ITEM AT INDEX ="; INDEX (CH);
210 INPUT %CH,A$
220 PRINT " DATA FOUND = "; A$
230 GOTO 140
240 INDEX <CH>=INDEX(CH)+1
250 GOTO 140
```

In this program, a match is being sought in field 2. The range of locations covered by field 2 is specified by the value of FP(2), which is the offset to the beginning location of the field and FL(2), the length of the field.

The program is explained as follows:

> Line 120 sets the INDEX equal to the beginning of the data file where the search begins.
>
> Line 140 orders the FIND on string S$ and specifies a channel CH.
>
> Line 150 tests for the string match anywhere in the file, including past the end of valid data; if a match is not found, a "Not Found" message is returned.
>
> Line 160 calculates offset from beginning of data in file to the location of the match.
>
> Line 170 calculates the relative offset to the beginning of the record the match is found in.
>
> Line 180 calculates how far into the record the match starts.
>
> Line 190 instructs that if a match occurs, but not in field 2, then go to line 240 which increments the INDEX by 1 and searches again starting with the first position after the last match.
>
> Line 200 instructs a print of the location where the match was found.
>
> Line 210 inputs the string found within the specified range.
>
> Line 220 prints the string found within the specified range.
>
> Line 240 increments the INDEX to the next location after the last match

KEYBASE FILES

An overview of the M/A-COM OSI Keybase system is given here as an example of Keyware OS-65U advanced file structures and programming techniques. Most of the program files in Keybase utilize an advanced programming technique called associative accessing. This means that the utility files are insensitive to changes in file size, amount and order of information in each file; data files used by the Keybase system are organized to provide them with the same degree of flexibility.

Keybase data files are organized for direct access similar to the Direct Field files described earlier in this chapter. The primary difference is the header placed at the beginning of the Keybase file and the way in which the INDEX is used to interface between Keybase master files and Keybase key files.

Master Files

Keybase master files contain the fundamental portion of the business data. As such, they act as the foundation for the user to build the business system; they also act as a basis for the programmer to create new applications. Figure 4-6 illustrates the structure of a typical Keybase master file.

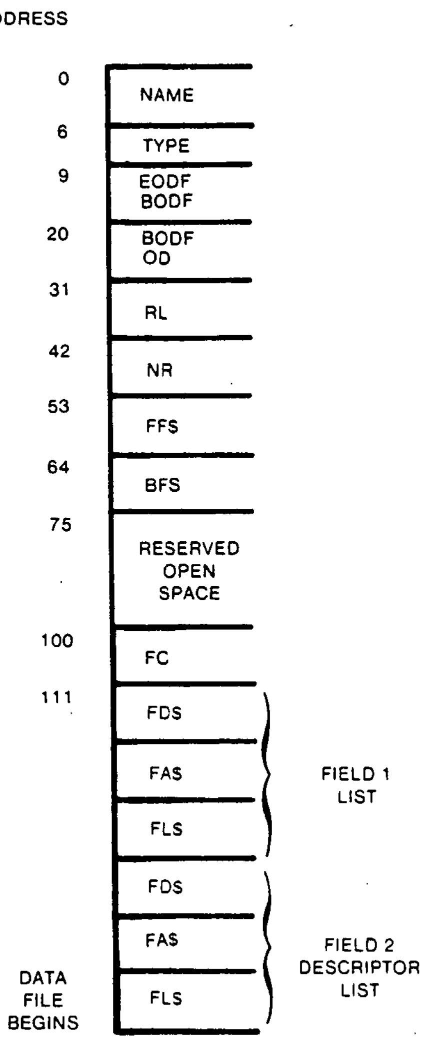

FIGURE 4-6

The master file establishes and utilizes a preset file header which can vary in length. The header fields are explained as follows:

NAME	The name of the file contained in first five characters. If file is less than five characters remaining locations are padded with blanks.
TYPE	File type. 20 indicates a master file. 21-27 indicate key files.
EODF	End of data file. This field provides a pointer to the logical end of the file, generally for the purpose of appending to it.
BODF	Beginning of data file. This provides a pointer to the location where data actually starts.
RL	Record length. Length of each records in the file.

RC	Record count. Indicates how many records there are room for in the file.
FC	Field count. Number of fields in each record.
FD$(n)	Field descriptor (string) for first field. Example: Company Name
FT$(n)	Field type for first field. Possibilities are: A,I,D,C,F,E (ASCII, Integer, Date, Cash, Floating Point or Encoded).
FL$(n)	Field length (string) for field n. Example: FL$(40) indicates field n (Name) is 40 bytes in length.

Key Files Keybase key files are user-defined direct access files with fixed length records that contain one entry for each record existing in the master file. Key files are based on a master file; they allow fast and easy access of data stored in a master file by locating information in the master file through use of the key file. Key files act as an aid to the master file in two major ways: First, through the use of an ISAM (Indexed Sequential Access Method) program, the key files provide a very fast sequential search of the master. Secondly, the key file can be sorted instead of the master. By doing so, the master file may be accessed in various ways via the key file. Also, sorting by key files instead of the master saves a good deal of time since the key files are much shorter. Figure 4-7 shows the structure of a key file.

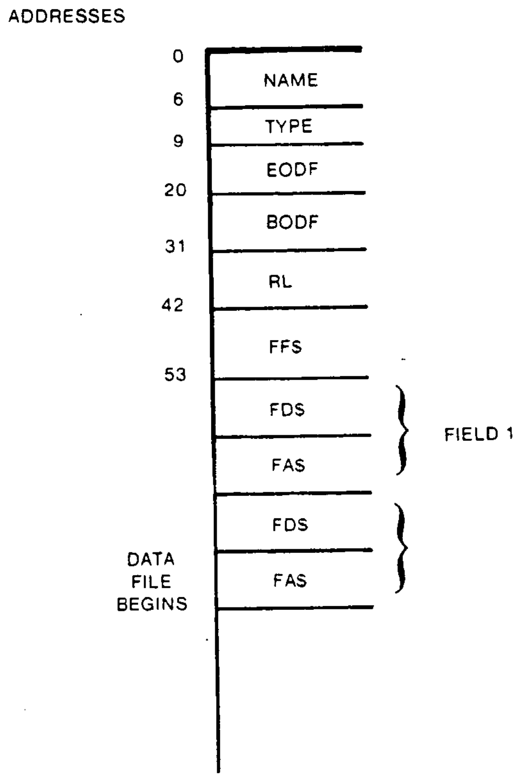

FIGURE 4-7

Definitions of the key file headers are listed below for reference. While key files have fewer headers than master files; the header definitions are the same.

NAME	The name of the file contained in first five characters. If file is less than five characters, remaining locations are padded.
TYPE	File type. Key file types end in numbers 1-7 such as 21-27.
EODF	End of data. This record provides a pointer to the logical end of the file, generally for the purpose of appending to it.
BODF	Beginning of data file. This provides a pointer to the location where data actually starts.
FD$(n)	Field descriptor (string) for field n. Example: Company Name
FL$(n)	Field length (string) for field n. Example: FL$(40) indicates the field n (Name) is 40 bytes in length.
FT$(n)	Field type for field n. Possibilities include A,I,D,C,F,E (ASCII, Integer, Date, Cash, Floating Point or Encoded).

Each entry in the key file corresponds to a record in the master file. When the key file was created, the field it corresponded to in the master was specified, creating a pointer into the master file. A key file loader program asks for the name of the master file and the key file; it then goes to the key file to determine which field it should be looking for. The field from each record in the master file is retrieved and each record's starting index calculated. Each field's contents are then printed along with the starting INDEX of that record in the master file. This process is outlined in Figure 4-8.

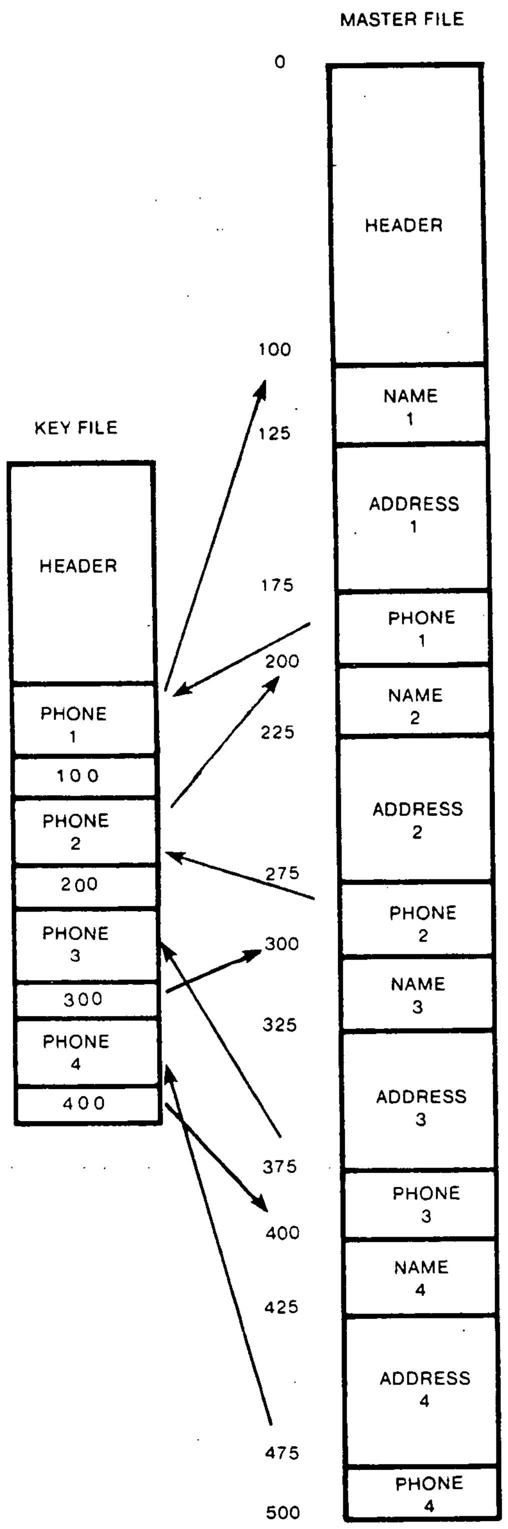
MASTER FILE
KEY FILE
0
100
125
175
200
225
275
300
325
375
400
425
475
500
HEADER
NAME 1
ADDRESS 1
PHONE 1
NAME 2
ADDRESS 2
PHONE 2
NAME 3
ADDRESS 3
PHONE 3
NAME 4
ADDRESS 4
PHONE 4
HEADER
PHONE 1
100
PHONE 2
200
PHONE 3
300
PHONE 4
400

FIGURE 4-8

Application programs can use a key file to quickly locate a record in the master file by using data in the key file, (such as company name). Each master file can have up to seven key files, which means it can be sorted/searched seven ways without modifying the actual master file.

FLAG COMMANDS

The Keyware OS-65U FLAG commands are used to enable and disable certain of the operating system's features. They are designed to provide assistance to the programmer in such areas as debugging, error-handling, file merging, input/output, etc. The FLAG commands operate much like switches -- they are either "on" or "off". They maintain their setting until it is reversed by a command from immediate mode or from within a program -- or a system reboot. FLAG commands that are automatically set upon system startup to create the necessary conditions for the system to operate properly have been identified.

The FLAG commands have been listed in two ways in the chapter: by number and in functional categories. The first list gives the purpose of each FLAG; the second categorizes the FLAG commands by the tasks they perform.

NUMERIC LIST OF FLAG COMMANDS

The syntax of the FLAG command is

FLAG n

where n is a FLAG number. A summary description of the FLAG is underscored for quick identification. For reference, the FLAGs are also listed in an Appendix with this summary description.

1. Disable flag 2.

2. Enable the close-files-on-error and the close-files-on-immediate-mode feature. With this feature enabled, if an error condition is encountered, or if the user enters immediate mode, any channels currently opened will automatically be closed. Before they are closed, any buffers pending to be written to disk will be written -- provided an error condition does not prevent it.

5. Enable user programmable disk end-of-file action. Sets the channel INDEX to a value recognized by the system as the end of the file upon encountering a disk file end-of-file condition.

CAUTION

If this FLAG is executed, it is the responsibility of the program to do all end-of-file checks before attempting an INPUT or PRINT. If disk input/output beyond the end of the file is attempted, the system may fail.

6. Disable FLAG 5. Enables program abort and system error message upon disk end-of-file condition.

7. Enable BASIC statement trace. With this feature enabled, each program line executed will be displayed on the console.

NOTE: During Tracing, the TAB command may not appear to work correctly during output to a printer.

8. Disable FLAG 7.

9. Enable user programmable disk error action. With this option enabled, program execution will jump to line 50000 upon encountering a disk error condition instead of forcing the user to immediate mode (FLAG 23 enables both disk and BASIC errors to be processed at line 50000).

10. Disable FLAG 9. Enables program abort and system error message upon encountering a disk error condition. This Flag is overrridden by FLAG 9 (FLAG 24 directs both disk and BASIC errors to the immediate mode).

11. Enable space suppression in numeric output to files. Suppresses spaces before and after all numbers (and prints minus signs before negative numbers).

12. Disable FLAG 11.

13. Enable "INPUT%n," command file operation. Using this option, it is possible to LIST a program or parts of a program to a data file and later INPUT this data file to another program. This provides a convenient way to merge programs. See the Merging BASIC Programs section of the "Programming Aids" chapter of this Guide for more detail.

14. Disable FLAG 13.

15. Allow the characters comma (","), and colon (":") to be treated as valid terminators in an INPUT or READ statement.

CAUTION

Use caution in executing READ statements with this FLAG enabled, since most READ statements have commas within them as delimiters.

16. Disable FLAG 15. Allows the comma and colon to be treated as INPUT or READ delimiters or terminators.

17. Disable carriage return/line feed upon terminating an INPUT or PRINT. This option is most often used with a formatted screen.

CAUTION

All I/O devices are affected by this command; all carriage return/line feed sequences are suppressed. The most advisable way to program this FLAG is in conjunction with FLAG 18 as follows:

```
FLAG 17          Turn off CR/LF
  INPUT A$
FLAG 18          Turn on CR/LF immediately
                 after INPUT
```

18. Disable FLAG 17. Enables carriage return/line feed upon terminating an INPUT or PRINT>

21. Disable FLAG 22. Disables input escape on carriage return. With this Flag enabled, if an operator enters just a carriage return for an INPUT statement, the system response will be a "REDO FROM START" message followed by a "?" prompt for re-entering the expected INPUT value. This Flag may be overridden by a FLAG 27.

22. Enable input escape on carriage return. With this Flag enabled, a response of a carriage return to an INPUT statement will cause entry into the immediate mode. This FLAG may be also overridden by FLAG 28.

23. Enable a jump to program line 50000 upon the occurrence of any or all error conditions, including BASIC and disk errors. FS (Full Stack) and OM (Out of Memory) errors CLEAR the variables for loop and GOSUB...RETURN stack before going to line 50000. If FLAG 23 is enabled and the program has no line 50000, automatic entry into immediate mode will result. This Flag may be overridden by FLAG 9.

24. Disable FLAG 23. Enables automatic entry into immediate mode upon the occurrence of any BASIC or disk error conditions.

25. Disable FLAG 26. Disables Control "C" termination of BASIC program execution.

26. Enable Control "C" termination of BASIC program execution.With this FLAG on, pressing the Control and 'C' keys will terminate a BASIC program during execution and cause entry into immediate mode.

27. Enable a null input (carriage return only) as valid INPUT sequence.

28. Disable FLAG 27. Disables null input sequence on INPUT. With this Flag on, a null input will cause entry the immediate mode, unless overridden by a FLAG 21.

29. Disable FLAG 30. Disables a trap-overflow condition.

30. Enable a trap-overflow condition. OS-65U BASIC's nine-digit precision allows perfect representation only up to 4,294,967,295. Numbers beyond that are rounded off and therefore not precisely represented. FLAG 30 generates An OV (Overflow) error for numbers greater than 4,294,967,295. This can be used as a debug feature for business programs that require detailed accuracy.

31. Disable FLAG 32. Disables input translation.

32. Enable input translation. Immediate mode input of lower case characters will be translated to upper case unless they are within quotes or in a REM statement. This FLAG allows commands such as LIST to work.

CAUTION

If lower case is to be used in DATA statements remember to enclose the string in quotes or it will be translated.

33. Disable FLAG 34. Disables output translation on LIST.

34. Enable output translation on LIST. All upper case characters are translated to lower case with the following exceptions: 1) Data with quotes or on a REM line. 2) All reserved words have the only the first letter capitalized -- except for REM which remains capitalized.

NOTE: The translation is only done on the characters output. The program source in memory and on the disk is unchanged.

100 Perform a conditional top-of-form eject on print device #**5**.
If page is at top-of-form, it will not be ejected. This is available only if the system paging feature is enabled (see Chapter 8, the "PRTMAP" section).

101 Disable FLAG 100. Unconditional top-of-form page eject. Availability and time sharing behavior the same as FLAG 100.

FLAGS AT SYSTEM START UP

The following is a list of FLAGs that are automatically set on system startup.

FLAG 1	Disable close-files on error or immediate mode entry.
FLAG 6	Enable abort and error message on disk end-of-file.
FLAG 8	Disable BASIC statement trace.
FLAG 10	Enable abort and message on disk error.
FLAG 12	Disable space suppression on numeric ouptut to file.
FLAG 14	Disable 'INPUT n,' command file operation.
FLAG 16	Disable colons and commas on INPUT.
FLAG 18	Enable carriage return/line feed on INPUT & PRINT.
FLAG 22	Disable input escape on carriage return.
FLAG 24	Go to immediate mode on all error conditions.
FLAG 25	Disable Control/C termination.
FLAG 27	Enable null on input.
FLAG 29	Disable trap of overflow error.
FLAG 31	Disable input translation.
FLAG 33	Disable output translation.

FUNCTIONAL CATEGORIES OF FLAG COMMANDS

The FLAG commands below are listed numerically in categories by tasks they perform. FLAGs that appear in more than one category are marked with an asterisk (*).

Debugging FLAGs

FLAG 7	Enable BASIC statement trace
FLAG 8	Disable FLAG 7
FLAG 30	Enable trap-overflow condition
FLAG 29	Disable FLAG 30

END-OF-FILE FLAGS

FLAG 5	Perform user-programmed action for end-of-file
FLAG 6	Disable FLAG 5. Aborts and prints message for end-of-file

Error-Handling FLAGs

*FLAG 2	Enable close-files-on-error
*FLAG 1	Disable FLAG 2
FLAG 9	Perform user-programmed action for disk errors
FLAG 10	Disable FLAG 9. Aborts and prints message for disk
FLAG 23	Perform user-programmed action for any error
FLAG 24	Disable FLAG 23. Goes to immediate mode for any error
FLAG 30	Enable trap-overflow condition
FLAG 29	Disable FLAG 30

Exit FLAGs

FLAG 26	Enable exit on Control-C.
FLAG 25	Disable FLAG 26

File-Closing FLAGs

*FLAG 2	Enable close-files-on-error and close-files-on-immediate mode.
*FLAG 1	Disable FLAG 2.

File-Merging FLAGs

FLAG 13	Enable "INPUT n," command file function
FLAG 14	Disable FLAG 13

Input FLAGS

FLAG 15	Treat comma and colon as characters
FLAG 16	Disable FLAG 15
*FLAG 18	Enable carriage return/line feed at end of INPUT
*FLAG 17	Disable FLAG 18.
FLAG 22	Enable input escapeon carriage return
FLAG 21	Disable FLAG 22.
FLAG 27	Accept null string in INPUT
FLAG 28	Disable FLAG 27
FLAG 32	Enable input translation
FLAG 31	Disable FLAG 32

Input/Output Conversion FLAGs

FLAG 11	Enable space suppression on numeric output and numeric-to-string conversions
FLAG 12	Disable FLAG 114
*FLAG 18	Enable carriage return
*FLAG 17	Disable FLAG 18
FLAG 34	Enable output translation on LIST
FLAG 33	Disable FLAG 34

Printer Control FLAGs

FLAG 100	Perform conditional top-of-form eject on print device #5
FLAG 101	Perform unconditional top-of-form eject on print device #5

THE TRANSIENT FUNCTIONS

This chapter describes a group of programs designed to allow the programmer to configure the OS-65U environment in which program development and execution occur. These programs are called Transient Functions and are listed on the Transient Functions menu. They provide the programmer with capabilities for editing and resequencing program lines, verifying INPUT and PRINT data, retaining variables between programs, configuring for terminal independence.

The interrelationship of the Transient Functions is indicated by Figure 6-1. Three of them, Editor, Extended Input and Common Variables are related, while the remaining two, Resequencer and Terminal Setup operate independently. It means that enabling or disabling Extended Input automatically enables or disables the features of the Editor, and enabling or disabling Common Variables automatically enable or disable the features of Extended Input and the Editor.

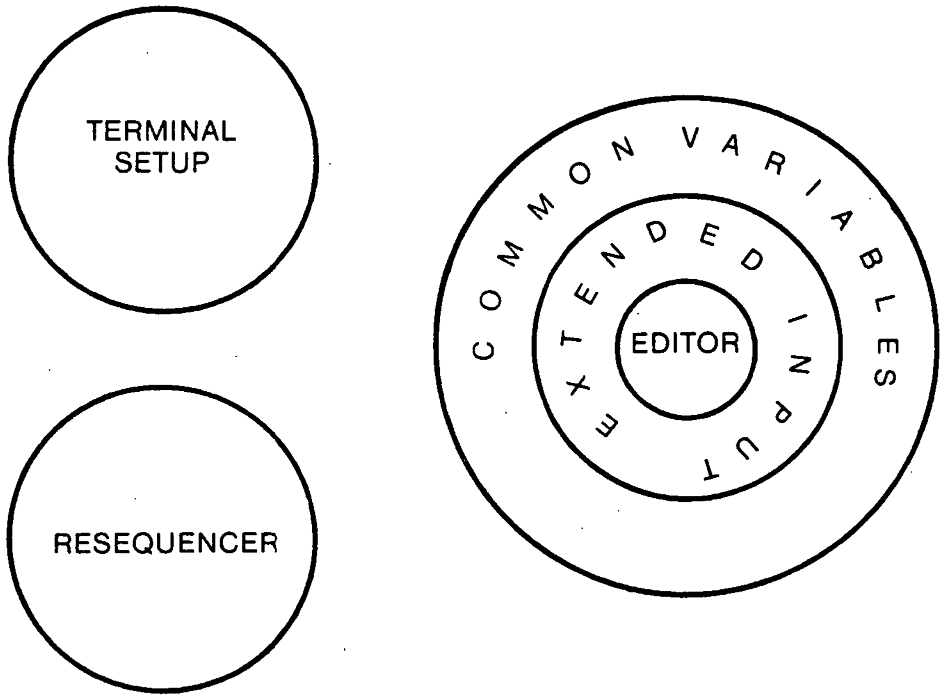

FIGURE 6-1

Since certain of the Transient Functions make some BASIC commands invalid, they should be used only by experienced programmers after they fully understand them. Also, the Transient Functions should always be enabled and disabled by menu selection; the menu system is designed to disable exclusive Transients before enabling others.

WARNING

Memory is cleared when a Transient Function is enabled. Be certain the file in the workspace has been SAVEd to disk.

The Transient Functions menu (listed below) is accessed by selecting it from the Main menu.

1) Editor
2) Resequencer
3) Extended Input
4) Common Variables
5) Terminal Setup
6) Standard System

The Transient Functions menu can also be reached directly from immediate mode with the command

```
RUN"/"
```

A listing and summary description of the Transient Functions is given in an Appendix.

THE LINE EDITOR (EDITOR)

EDITOR is an extension to the operating system that provides on-line editing capabilities to program lines. When enabled, it associates to the last file that was loaded into memory. It is enabled by selecting "Editor" from the Transient Functions menu. When enabled, the statement

```
Editor currently enabled
```

appears above the menu.

The Editor is terminal independent, in that any user may configure a specified CRT terminal to use it. For example, in a timesharing environment, users could be working on different types of terminals and all use the Editor. This is explained in more detail in the "Terminal Setup" section later in this chapter.

As previously mentioned, the Editor's features are automatically enabled with the Extended Input and the Common Variables Functions. Those programs are discussed later in this chapter. When enabled with another Transient, the Editor's presence is not indicated to the user and cannot be disabled independently. However, the Editor capabilities available to the user are identical whether it has been enabled independently or with another Transient.

Editor Functions

Following is a summary of the operator-entered control sequences for editing and cursor manipulation made available by the Editor:

!nnnn	Recalls program line number nnnn for editing; for example, !1055 or !12. An attempt to recall a non-existant line number returns the next higher one. If no higher one exists, it results in a <cr> and line feed.
!	Recalls the next sequentially available line for editing-- after the starting line has been defined using the previous sequence.
!!	Recalls the same line again for re-editing.
@ ("at" sign)	Cancels all edits entered for a line. The command erases the current line from the screen, but leaves the (original) line in memory.
Control F	Moves cursor to the front of the line.
Control R	Moves cursor the the rear of the line.
Control L	Moves cursor one space forward (right).
Control H	Moves cursor one space back (left).
RUBOUT or DELETE	Deletes character at cursor position; if at end of line, deletes previous character.
Control I or TAB	Tabs eight character positions to the right. Cannot tab beyond the end of the line.
Control T	Toggles (alternates) between character insert and overstrike mode. In the insert mode, each character keyed is inserted into the line at the cursor; in the overstrike mode, the character entered replaces the character at the cursor.

Disabling the Editor

Use one of the following procedures to disable the Editor:

1. Run the Standard System selection from the Transient Functions menu. This option will disable any Transient and return the system to standard configuration.

2. Enable the Resequencer from the Transient Functions menu.

THE RESEQUENCER

The Resequencer allows the programmer to renumber BASIC program lines. It allows changes to be specified for

- The starting line number in the program.
- The starting line number in the newly numbered program.
- The line number increment value.

Enabling the Resequencer

The Resequencer is enabled by selecting it from the Transient Functions menu. Once enabled, the statement

The Resequencer is enabled

appears above the menu. With the Resequencer enabled, all other Transient Functions are disabled.

Using the Resequencer

Once enabled, the Resequencer can be used as a BASIC command. Its syntax is:

```
RSEQ NL,OL,IN
```

where the options are defined as

NL	=	Starting new line number
OL	=	Old line number at which to begin resequencing
IN	=	Line increment

Each of the options has a default. They are as follows:

NL	=	First line in the program
OL	=	First line in the program
IN	=	Line increment of ten

To have the default values recognized, delimiting commas must be entered adjacent to ommitted (defaulted) options. If all three options are to be defaulted, enter only

```
RSEQ
```

with no options specified.

Example

To resequence a program starting with new line 50, beginning with the first line of the old program and an increment of 100, the command is:

```
RSEQ 50,,100
```

Note that commas were entered for the second option, in order to use the default.

Disabling the Resequencer

Use one of the following methods to disable the Resequencer.

1. Select the Standard System option from the Transient Functions menu.

2. Enable any of the other Transient Functions from the menu.

EXTENDED INPUT

The Extended Input Transient Function is an operating system extension that gives the programmer the benefits of extended INPUT and PRINT statements. This feature provides for greater verification of data by providing the programmer with the means of

1) Limiting the input character set to that which was selected.

2) Verifying that the format of data is valid for input.

3) Requiring invalid input to be corrected through use of the Editor.

Extended Input automatically enables the features of the Editor. Therefore, besides its own capabilities, this Transient Function provides full line editing.

Extended Input is an overlay to standard Keyware 65U BASIC and modifies some of its features. With Extended Input enabled, the following BASIC commands are no longer available. Attempts to use them will produce a syntax (SN) error.

LOG	SIN	TAN
SQR	RND	ATN
EXP	COS	(exponential operator)

The INPUT Statement

Extended Input allows the programmer to control the INPUT statement more completely that regular BASIC by allowing data types and field lengths to be specified.

With Extended Input enabled, the syntax for the BASIC INPUT command is as follows:

```
INPUT %n, [L,T$] QA$                    n=channel number
INPUT #n, [L,T$] QA$                    n=device number
INPUT "prompt string"; [L,T$] QA$
```

where

L = maximum length of the string to be input. L must evaluate to an integer between 0-254.

T$ data type of the string. Types currently defined are:

A - ASCII. This form allows input of any legal ASCII character, with leading spaces maintained, i.e., not thrown away as in normal BASIC input.

E - Encoded. User-defineable field type with contents treated as ASCII. Used to input encoded user data. An example would be a Payroll application where Name/Address fields could be changed but not dollar amounts. Leading spaces are maintained.

D - Date. Used to preserve date exactly as in file. Used to distinguish date field from ASCII field in order to permit program conversions that may be necessary. Input allowed exactly as ASCII. Leading spaces are maintained.

I - Integer. This form accepts characters "0" through "9", "+", and ""-" only. Leading spaces are discarded.

C - Cash. This form accepts a real number in the form nnn.nn, i.e., a real number with two digits to the right of the decimal point if a decimal is entered. Leading spaces are discarded.

F - Floating point. Same as Cash (above), but any number of digits to the right of the decimal is allowed. Leading spaces are discarded.

QA$ - preloaded string to be INPUT. The original contents of this string, if any, will be displayed on the CRT at the time of INPUT. Extended Input requires string (non-numeric) input.

Using the above form of INPUT command, INPUT strings may be up to 254 characters in length maximum.

Example of Extended INPUT Statement

Assume the following INPUT statement were entered with Extended Input enabled:

```
QA$="1.23"                        :REM preloaded value
PRINT"Enter cash value";
INPUT [6,"C"]QA$                  :REM cash field, 6 chars max
```

The operator will see "Enter cash value 1.23" on the console, with the cursor positioned on the "1" character. Should the operator enter an illegal character such as a "Q" for instance, the bell will sound and the entry will not be accepted or echoed.

Assume the operator is in the Editor's insert mode (see the "Control T" and other Editor functions described earlier in this chapter). The operator now presses the "8" key, and "81.23" appears on the console. After pressing <RETURN>, the value of QA$ after INPUT is "81.23".

The operator now enters "@" ("at" sign -- Editor function) to delete the QA$ entry. At that point, the operator will see "Enter cash value " with the cursor to the immediate right of "value". The operator enters "5.23.4". Since this is an illegal INPUT entry for Cash, the operator is alerted by the console bell and the cursor is positioned on the rightmost decimal. The operator deletes this decimal with the

(Editor's) RUBOUT or DELETE key and "5.234" is displayed on the screen. The entry is still illegal since it contains three digits to the right of the decimal point. The bell sounds and the cursor appears over the "4" character. The operator deletes this character and INPUT is terminated with a new QA$ value of "5.23".

The PRINT Statement

Extended Input provides an extension of the BASIC PRINT statement to allow for field length as well as left/right justification of the data entered. With Extended Input enabled, syntax for the PRINT command is as follows:

```
PRINT [FL-1,J$] QA$

PRINT#N,[FL-1,J$] QA$            N=device number

PRINT%CH,[FL-1,J$] B$            CH=channel number
```

where

FL-1	=	print field length minus carriage return
J$	=	"L" for left justified
		"R" for right justified

Since the PRINT statement parameters may be any combination of variables or literals, the following are all valid PRINT statements with Extended Input enabled.

```
PRINT [3,TY$] QA$                PRINT [FL,"R"] QA$
PRINT [3,"L"] QA$                PRINT [3,"R"] "1000-101"
PRINT#DV,[FL,"L"] QA$            PRINT%CH,[FL,"R"] QA$
PRINT#5,[FL,"L"] QA$             PRINT%1,[10,"R"] "HELLO"

PRINT #DV,[10,"R"]'10202";X;TAB(45);[20,"R"]H$
```

As an example, to direct the string "   XXYY   " to the line printer, the following PRINT statement is required:

```
PRINT#5,[5,"R"]"XX;[4,"L"]"YY"
```

NOTE: If the length of the string to be printed is greater than the specified field width, a long string (LS) error results.

To right/left justify PRINT data, a pad (filler) character is necessary to take up space. The PRINT pad character is normally a space (ASCII 32) and is available only when Extended Input is enabled. If it is desirable to change this character, it can be POKEd into memory location 12098. For example, to change the filler character to an asterisk, the following command would be used:

```
POKE 12098,ASC("*")
```

The Extended Input PRINT is preferable to use over the regular PRINT statement because the regular form requires a concatenation of more than one string variable. This concatenation of strings generates "garbage" in the workspace which the system will eventually have to stop and clean up, resulting in a delay.

Disabling Extended Input

Extended Input is disabled by the program INPOUT. Either of the procedures listed below invoke INPOUT.

1. Select Standard System from the Transient Functions menu.
2. Enable a mutually exclusive Transient Function from the menu.

THE COMMON VARIABLES MODE

Common Variables mode is an extension to the operating system that gives the user the ability to save variable values (the entire variable workspace) upon RUNning a program from another program. This feature provides the ability to chain programs together for a 'virtual' effect to run larger programs than can actually fit in the memory workspace. With Common Variables enabled, the user has the option of RUNning programs and retaining variables or not retaining them. Also, with Common Variables enabled, the BASIC command NULL is replaced with KILL; this command deletes the variables' definition and frees up memory space.

The presence of the Common Variables mode can be detected while in the BASIC immediate mode. If the screen prompt characters are "OK" (upper case), Common Variables is not enabled; if the prompt characters are lower case "ok", it is enabled.

The Common Variables mode is an overlay to BASIC, and the following reserved words are not available when it is enabled:

FNA   DEF   NULL

With Common Variables enabled, the RUN command syntax that will retain the variable workspace values for use by the next program is

RUN[PR$,PW$,LN]

where PW$ is the program password and LN is the line number at which to start execution. The standard BASIC command RUN will not retain these variables.

Example Using Common Variables

Following is an example of running programs using Common Variables:

```
10 REM PROGA                      10 REM PROGB
20 X=100                          20 PRINT X
30 RUN ["PROGB","PASS",10]        30 END
40 END
```

First, the operator RUNs PROGA with the command RUN"PROGA". Line 30 will RUN PROGB, retaining the values of variables in PROGA, the value of X which is 100. In PROGB, the printed value of X will still be 100, as retained from PROGA. Should it be necessary to return to the original program (PROGA), a RUN[ ] statement containing the values of PROGA would have to be entered into PROGB.

The KILL Command

As mentioned previously, Common Variables replaces the NULL command with KILL to provide the capability of eliminating variables. The syntax and functions of the KILL command are as below.

KILL A,B,I, ...P$,...	Kills specific simple variables.
KILL A( ),B( ),P$( )	Kills specific array variables.
KILL *	Kills all simple variables.
KILL (*)	Kills all array variables.

CAUTION

NEVER use the KILL command from within a FOR-NEXT loop. Doing so has disastrous effects on the program.

After KILLing an array variable, it is possible to reDIMension that variable again. As an example:

```
DIM A(20,20)
:
:
:
KILL A()                 At this point, the array A no longer
:                        has any DIMension or value.
:
:
DIM A(100,100)
```

Disabling Common Variables

Common Variables may be disabled by either of the following procedures:

1. Select the Standard System option from the Transient Functions menu.
2. Enable any other Transient Function from the menu.

TERMINAL SETUP

The Terminal Setup program is used to configure the system for a specific CRT terminal so the terminal will be recognized by the line editor and cursor control functions. The parameters for the CRT terminals are stored in the Keybase master file named "CRT 0" which will be described later in this section.

Terminal Setup is selected from the Transient Functions menu. A list of currently defined terminals is produced. To make one of these terminals the system default, simply select it and press <RETURN>. This will move the selected terminal and its parameters to the first record (#1) in the "CRT 0" file. The required operating system functions are then able to access the terminal parameters from there.

Adding a terminal to the Terminal Setup list or changing the parameters of one of the terminals is done by modifying the "CRT 0" file. To do this, go to the System Utilities menu. Select the "General Purpose Keybase Compatible File Editor". This selects the program "ED". Respond as follows to the program prompts:

PROMPT	RESPONSE
DEVice?	A (System drive name)
File name,password?	CRT,PASS

The following menu will then appear:

1) Edit file by record number
2) Append to file
3) Print file or specific record on printer
4) Exit?

Editing The CRT 0 FILE

To edit an existing record in the CRT 0 file, choose "Edit file by record number". This will produce the prompt

Record number of record to edit.

which requests the terminal (by number) on the Terminal Setup list; each represents a separate record in the "CRT 0" file. The first record (number 1) of this file will be blank. The terminal selected will occupy record number 1 of the CRT 0 file; this record is used by the program to establish the system default. Therefore, the active record numbers actually begin with number 2. In other words, selecting CRT 0 record number 2 will produce the record for terminal number 1 on the Terminal Setup list.

Terminal name
Code recognized as incoming forward space command.
Code recognized as incoming backspace command.
Code(s) to be echoed to cause a forward space.
Code(s) to be echoed to cause a back space.
Code(s) to be echoed to address the cursor.
Code(s) to be echoed to clear the screen.
Code(s) to be echoed to clear to the end of the screen.
Code(s) to be echoed to clear to the end of the line.
Code(s) to be echoed to set foreground.
Code(s) to be echoed to set background.

The codes required are ASCII values (decimal) and should be obtainable from the terminal user manual. Note that encoding the commands properly is very important for the terminal to work properly. If your terminal does not appear on the Terminal Setup list, see Appendix F of this manual. If you still encounter difficulties, in contact your dealer or M/A-COM OSI.

The ED2 Program

Another editor, "ED2", may be used to edit the CRT 0 file (or any data file). It is run from immediate mode on hard disk or floppy diskette systems. To run it from floppy diskette systems, it is necessary that the following OS-65U programs be on one device:

```
ED2$
REFRDH$
GETCRT$
```

It is also necessary that all these programs remain on-line while the editor program (ED2) is running.

The ED2 program can be used to edit the "CRT 0" file in the same way as the "ED" program described here, except that it does it on a formmated screen. To invoke the program enter

```
RUN"ED2$
```

from immediate mode.

Example CRT 0 File Record

Following are the fields as filled in for a sample record #2 in the CRT 0 file.

```
2:1   TERMINAL NAME
        DEC VT100

2:2   IN FORWARD SPACE
        012

2:3   IN BACK SPACE
        008

2:4   ECHO FORWARD SPACE
        027 091 068 000

2:5   ECHO BACK SPACE
        027 091 068 000

2:6   ADDRESS CURSOR
        155 091 187 200 000 129 001

2:7   CLEAR SCREEN
        027 091 001 059 001 200 027 091 050 074 000

2:8   CLEAR TO END OF SCREEN
        027 091 074 000

2:9   CLEAR TO END OF LINE
        027 091 075 000

2:10  FOREGROUND
        027 091 052 109 000

2:11  BACKGROUND
        027 091 048 109 000
```

These codes are used by the line Editor where applicable and can be used in application programs for CRT screen manipulation. A sample use of the codes as used in a program can be found in the Chapter Seven of this guide under "Retrieving CRT Control Codes (GETCRT)".

Append To File

To add a terminal to the Terminal Setup list, use the procedures described in the previous section to open the "CRT 0" file. When the menu appears, select "Append to file". This will produce the next available open record. After the fields are filled in, the record is added to the file.

Disabling Terminal Setup

To disable Terminal Setup, enable Standard System or any other Transient Function from the menu.

STANDARD SYSTEM

This option disables any of the Transient Functions that may be enabled and reconfigures the system to standard KEYWARE 65U.

PROGRAMMING AIDS

This chapter covers Keyware OS-65U features that provide additional assistance to the programmer in writing and modifying programs. System characters for controlling the console are identified; also, features are described that enable the programmer to address and manipulate the cursor, recover errors, merge programs, modify memory size and debug programs. The last section of the chapter describes two programs and a FLAG command designed to provide assistance in debugging programs.

CONSOLE CONTROL CHARACTERS

Keyware OS-65U provides a number of control character commands for controlling output to the console and for executing BASIC. They are listed and defined below.

Control C:	Stops a BASIC program or execution at the end of the current statement if this option has been enabled with FLAG 26.
Control S:	Stops all output pending input of a Control Q.
Control Q:	Restarts output that was stopped with a Control S or Control D.
Control O:	Causes output to be "thrown away" pending input of another Control O or entry into the immediate mode.
Control D:	Limits output to one screen at a time, then stops pending input of a Control Q (console paging). This is only effective with output to the console.
Control W:	Terminates the paging of output that was initiated by a Control D.

HIGHER PRECISION ARITHMETIC

Some application programs require a higher degree of precision than is normally available with standard BASIC. It is possible to achieve this precision by means of the techniques described in this section.

Double Precision Accumulator Subroutine

OS-65U provides 9-digit precision BASIC. Numbers greater than 9 digits are rounded off and represented as accurately as possible using 9 digits. This limitation may be inadequate in some financial or other applications that need more precision for some calculations.

The following program provides addition and subtraction with 16 digit precision.

```
10 REM  DOUBLE PRECISION ACCUMULATOR TEST PROGRAM
20 REM
30 REM
40 SL=0: SH=0: REM INITIALIZE DOUBLE PRECISION ACCUMU-LATOR TO ZERO
50 INPUT X:    REM GET NEXT TEST VALUE FROM OPERATOR
60 GOSUB 100:  REM CALL DOUBLE PREC ACCUMULATE SUBR
70 PRINT ,S$:  REM PRINT CURRENT SUM
80 GOTO 50:    REM DO IT AGAIN
90 REM
100 REM  DOUBLE PRECISION ACCUMULATION SUBROUTINE
110 REM
120 REM ENTRY: X  = VALUE(+-) TO BE ADDED TO DOUBLE PREC SUM
130 REM EXIT:  S$ = FORMATTED VALUE OF SUM
140 REM
150 MAX=100 000 000: REM MAX VALUE IN LEAST SIGNIFICANT HALF OF ACCUM
160 REM
170 IF ABS(X)>=MAX THEN PRINT "INPUT TOO BIG": RETURN
180 REM
190 SO=SGN(SL): IF SO=0 THEN SO=SGN(SH): REM  SAVE SIGN OF OLD SUM
200 SL=SL+X:      REM  ACCUMULATE SUM
210 REM
220 REM -- ADJUST IF OVERFLOW
230 IF ABS(SL)>=MAX THEN SH=SH+SGN(SL): SL=SL-SGN(SL)*MAX: GOTO 280
240 REM
250 REM--ADJUST IF UNDERFLOW
260 IF SO*SGN(SL)<0 AND SH< >0 THEN SH=SH+SGN(X): SL=SL=SO*MAX
270 REM
280 REM  FORMAT SINGLE OUTPUT VALUE FROM DOUBLE PREC VALUE
290 IF SH=0 THEN S$=STR$(SL): GOTO 320
300 S$=STR$(SH)+MID$(STR$(SL+SGN(SH)*MAX),3)
310 REM
320 RETURN
```

The above program consists of two parts:

1) A test routine (lines 10 - 90) repeatedly requests input of a positive or negative number from the console operator. It accumulates the number in double precision and prints the current value of the accumulation. This test routine should not be included in an application program--with the exception of line 40; it must be included in programs using the double precision routine because its statements reset the value of the double

precision accumulation to zero. The entire routine is provided here only for use in evaluating the double precision accumulation subroutine.

2) The double precision accumulation subroutine (lines 100 - 320) should be copied explicitly into all application programs requiring double precision. If space in the program does not allow for all of it, the REMark statements can be omitted.

This double precision accumulation subroutine provides one of the best available methods for achieving high precision in OS-65U application programs. The approach was chosen from several alternatives because 1) It can be used in just those places where higher precision is required, and 2) It requires very little memory. Some of the other alternatives that are available are outlined below.

1. The BASIC could be changed to provide 16-digit precision. This would be simple to use, but would require more storage for each variable (8 bytes instead of 5), and all Floating Point operations would run significantly slower.

2. Application developers could develop or purchase an add-on mathematics package. These are significantly slower in execution and take much memory from the user workspace.

3. A dual precision (8 and 16-digit) option could be added to BASIC. This would provide a simple solution for writing new application programs, since 9-digit or 16-digit precision could be specified for each variable. However, existing programs requiring 16-digit precision would have to be modified. Also, BASIC would become much larger, resulting in an equivalent reduction in user workspace.

BINARY ROUND OFF ERROR

A problem frequently encountered in financial applications is that of 'binary round off error'. This error occurs because some fractional values cannot be perfectly represented in the binary form used internally by BASIC. This error can be avoided by carrying all monetary amounts in pennies instead of dollars/fractional dollars (cents). Immediately after input, a dollar and cents value should be converted to pennies as follows

```
P = INT(D*100+0.5)
```

All internal operations should be performed with the values expressed as pennies. When output, penny values should be converted back to dollars and cents as follows

```
PRINT $R,P/100
```

Using this approach, values up to 4,200,000,000 pennies or 42 million dollars can be accumulated error free in 9-digit BASIC. If larger accumulations are needed, the above double precision subroutine described earlier in this section should be used.

RETRIEVING CRT CONTROL CODES (GETCRT)

The GETCRT program is used to set up the required variables for using CRT functions and addressing the cursor in BASIC programs. In the last chapter, the Common Variables and Extended Input transient functions were described, as well as the "CRT 0" file which is maintained by the Terminal Setup transient. The Transient Functions are described in Chapter Six; be certain to read the chapter and understand the Transients before using the GETCRT program.

Two approaches may be used in programming with GETCRT. The first method described is with Common Variables enabled, the second with Extended Input enabled.

GETCRT With Common Variables Enabled

The Common Variables function, when enabled, uses the codes in record number 1 of the "CRT 0" file as the system default; it retrieves the codes from the file, encodes them and places them into memory. A subroutine is then required to address the cursor. Values are then established that will identify the user program that GETCRT is to return to. When GETCRT is run, the Common Variables utility will then retain the values while the program calls the Retrieve subroutine from disk and returns to the user program at the line specified. The steps are outlined below followed by an example.

1) Enable the Common Variables mode.

2) The subroutine to address the cursor must be incorporated into the user program. This can be done by direct entry as described here or by merging programs with FLAG 13 as described later in this chapter under "Merging BASIC Programs".

3) The values to identify the user program that GETCRT will return to should now be established. They are:

PR$	=	(Program name)
PW$	=	(Password)
LN	=	(Line number)

4) Set up the user program to RUN GETCRT with Common Variables enabled.

5) With the above conditions fulfilled, the program may be written, using the following statements where required:

PRINT CS$	=	(Clear screen)
PRINT CE$	=	(Clear to end of screen)
PRINT CL$	=	(Clear to end of line)
PRINT FG$	=	(Foreground)
PRINT BG$	=	(Background)
PRINT BL$	=	(Ring bell)

6) To address the cursor, values must be established for the X and Y coordinates and for the subroutine to address the cursor.

Sample GETCRT Program With Common Variables Enabled

Listed below is a sample program "MYPROG" using GETCRT with Common Variables enabled.

```
 10 PR$="MYPROG":PW$="PASS":LN=30
 20 RUN ["GETCRT","PASS",10]
 30 GOTO 300
         :
         :
100 POKE 22,X:ON AR GOTO 101,102,103,103:REM Address Cursor
101 PRINT AD$;CHR$(X+XF);DL$;CHR$(Y+YF);DE$;:RETURN
102 PRINT AD$;CHR$(Y+XF);DL$;CHR$(X+XF);DE$;:RETURN
103 X$+MID$(STR$(X+100)):Y$=MID$(STR$(Y+100+YF)
104 IF AR=3 THEN PRINT AD$;X$;DL$;Y$;DE$;:RETURN
105 PRINT AD$;Y$;DL$;X$;DE$;:RETURN
         :
         :
300 X+10:Y+12:GOSUB 100
310 INPUT "AMOUNT";QA$
```

In this sample, Line 10 establishes the program name password and line number that GETCRT is to return to, Line 20 invokes GETCRT to run the Retrieve subroutine and return to Line 30. Line 300 sets the cursor X and Y coordinates and runs the ADdress Cursor subroutine at line 100. Line 310 displays the prompt "AMOUNT?" with the cursor positioned at the coordinates specified.

GETCRT With Extended Input Enabled

Similar to the Common Variables Transient Function, Extended Input, when enabled, uses the codes in record number 1 of the "CRT 0" file as the system default by retrieving them from the file, encoding them and placing them into memory. With Extended Input enabled, two GETCRT subroutines are required -- to decode the CRT codes, and to set up the necessary strings for cursor addressing. Although more subroutine entry is required with the Extended Input approach, in certain applications it has advantages over the Common Variables approach because the program will involve fewer disk transfers -- a factor that can be important, especially when programming for multi-user systems. The steps for using the GETCRT program with Extended Input are outlined below, followed by an example program.

1) All programming is done with Extended Input enabled.

2) The GETCRT subroutines for retrieving the CRT codes and addressing the cursor (listed below) must be incorporated into the program at the beginning line numbers. This may be done by entering the subroutines manually or can be done with FLAG 13 as described later in this chapter section titled "Merging BASIC Programs".

3) With the above conditions fulfilled, the program may be written, using the following statements where required:

PRINT CS$	=	(Clear screen)
PRINT CE$	=	(Clear to end of screen)
PRINT CL$	=	(Clear to end of line)
PRINT FG$	=	(Foreground)
PRINT BG$	=	(Background)
PRINT BL$	=	(Ring bell)

4) To address the cursor, values must be established for the X and Y coordinates and for the Address Cursor subroutine.

Sample GETCRT Program With Extended Input Enabled

Listed below is a skeleton sample program invoking GETCRT with Extended Input enabled.

```
   10 GOSUB 63900
   20 GOTO 200  :REM JUMP TO FIRST USER STATEMENT
       :
       :
  100 POKE 22,X:ON AR GOTO 101,102,103,103:REM Address Cursor
  101 PRINT AD$;CHR$(X+XF);DL$;CHR$(Y+YF);DE$;:RETURN
  102 PRINT AD$;CHR$(Y+XF);DL$;CHR$(X+XF);DE$;:RETURN
  103 X$+MID$(STR$(X+100)):Y$=MID$(STR$(Y+100+YF))
  104 IF AR=3 THEN PRINT AD$;X$;DL$;Y$;DE$;:RETURN
  105 PRINT AD$;Y$;DL$;X$;DE$;:RETURN
       :
       :
  200 REM BEGIN USER PROGRAM HERE
       :
       :
  300 X=10:Y=12:GOSUB 100
  310 INPUT "AMOUNT";QA$
       :
       :
63900 Z=6345:AD=100:AD$="":DL$="":DE$="":AR=1:XF=0:YF=0
63904 Z1=PEEK(Z):REM - Address Cursor -
63905 IF Z1>127 THEN AR=2:Z1-128:REM Determine (x,y) Order
63906 AD$=AD$+CHR$(Z1)::REM Adr Cur Leadin
63907 Z=Z+1:Z1=PEEK(Z)
63908 IF Z1<128 AND Z1< >0 GOTO 63906
63909 IF Z1=0 GOTO 63915
63910 Z1=Z1-128
63911 DL$=DL$+CHR$(Z1):REM Adr Cur Delimiter
63912 Z=Z+1:Z1=PEEK(Z)
63913 IF Z<128 AND Z< >0 GOTO 63911
63914 IF Z1=0 GOTO 63915
63915 DE$="":GOTO 63917:REM Adr Cur Ending Delimiter
63916 Z=Z+1:Z1=PEEK(Z)
63917 IF Z1< >0 THEN DE$=DE$+CHR$(Z1):GOTO 63916
```

```
63918 XF=PEEK(Z+1):YF=PEEK(Z+2):REM Adr Cur Offsets
63919 IF XF>127 THEN XF=XF-128:AR=AR+2:REM Binary Ascii Flag
63920 Z=Z+3:CS$="":REM - Clr Scr -
63921 Z1=PEEK(Z):Z=Z+1:IF Z1< >0 THEN CS$=CS$+CHR$(Z1):GOTO 63921
63922 CS$=CS$+CHR$(13)
63923 CE$="":REM - Clr to End of Scr -
63924 Z1=PEEK(Z):Z=Z+1:IF Z1< >0 THEN CE$=CE$+CHR$(Z1):GOTO 63924
63925 CL$="":REM - Clr to End of Line -
63926 Z1=PEEK(Z):Z=Z+1:IF Z1< >0 THEN CL$=CL$+CHR$(Z1):GOTO 63926
63927 FG$="":REM - Foreground -
63928 Z1=PEEK(Z):Z=Z+1:IF Z1< >0 THEN FG$=FG$+CHR$(Z1):GOTO 63928
63929 BG$="":REM - Background -
63930 Z1=PEEK(Z):Z=Z+1:IF Z1< >0 THEN BG$=BG$+CHR$(Z1):GOTO 63930
63931 BL$=CHR$(7):RETURN
```

Line 10 of the sample calls the Retrieve subroutine, line 300 sets X and Y coordinates for the cursor and runs the Address Cursor subroutine at lines 100-105. Line 310 will display the "AMOUNT?" prompt with the cursor at the position specified by the coordinates.

USER PROGRAMMABLE ERROR RECOVERY

Keyware OS-65U provides the capability to send both disk errors and BASIC errors to line 50000. Most of the OS-65U utility programs use this error trapping method. In the absence of some form of error trapping, an error forces the user into BASIC immediate mode; this result is not always desirable for application programs.

Error trapping requires the execution of either FLAG 9 or FLAG 23. FLAG 9 will trap (send to line 50000) disk errors only. FLAG 23 will trap both disk and BASIC errors. A more complete discussion of FLAG commands is given in Chapter Five of this manual.

The following routine is used in conjunction with one of the FLAG commands mentioned above to decode errors. Note that this procedure just traps the errors; disposition of errors is the responsibility of the application program.

```
50000 EL=PEEK(11774)+256*PEEK(11775):REM Get Error Line
50010 EN=PEEK(18176):IF EN=23 GOTO 50100:REM BASIC or Disk
50018 :
50019 REM Decode BASIC Error
50020 Z$=CHR$(PEEK(EN+867)AND127)+CHR$(PEEK(868+EN)AND127)
50030 ER$="BASIC"+Z$+" Error in line"+STR$(EL)
50040 GOTO 50200
50098 :
50099 REM Decode Disk Error
50100 EN=PEEK(10226)
50110 Z=PEEK(9832):IF Z>127 THEN Z=Z-124:IF Z>63 THEN Z=Z-58
50120 ER$="Device "+CHR$(65+Z)" Disk Error"+STR$(EN)
50130 ER$=ER$+" in line"=STR$(EL)
50199 :
50200 PRINT:PRINT ER$:PRINT
```

Line 50000 retrieves the BASIC line number where the error occurred. Line 50010 distinguishes between a BASIC and Disk error. Lines 50020 - 50040 construct the error message for a BASIC error in the format: "BASIC XX Error in line XXXX". Lines 50100 - 50130 construct the error message for Disk errors in the format: "Disk error XX in line XXXX". Line 50200 prints the error message.

MAINTAINING SYSTEM DATE

Keyware OS-65U has the capability of recording a system date in the operating system. The locations for single user or intelligent work stations are listed below. Refer to the chapter on Timesharing for its date/time locations.

Day	24569
Month	24570
Year	24571

For the dates in these locations to be correct, a set up program must POKE them. A sample program is listed below.

```
10 INPUT "DAY, MONTH, YEAR";D,M,Y
20 POKE 24569,D
30 POKE 24570,M
40 POKE 24571,Y
50 PRINT "CURRENT DATE";
60 PRINT PEEK (24569);"/";
70 PRINT PEEK (24570);"/";
80 PRINT PEEK (24571);"/";
90 END
```

DETERMINING CURRENT SYSTEM

Keyware OS-65U allows for a simple method of determining the system level. One byte has been reserved at location 16317 for this purpose which may be PEEKed:

LV = PEEK(16317)

LV = System Level

0 = A release of 65U prior to version 1.3.
1 = Single User
2 = Network Intelligent Terminal
3 = Timesharing
4 = Timesharing with Network Extension

CAUTION

This location is set by the operating system and system programs. Application programs should never POKE into it. Doing so will have disastrous effects on the operating system.

It is helpful and often necessary to know if any of the Transient Functions in the system are enabled. Each Transient makes certain features available to the programmer; also some Transients disable certain BASIC commands (see Chapter Six). A PEEK at location 18959 indicates which, if any, are enabled.

```
TT = PEEK(18959)
```

0	=	No Transient Enabled
1	=	Editor Enabled
2	=	Resequencer Enabled
3	=	Extended Input Enabled (includes Editor)
4	=	Common Variables Enabled (includes Extend Input and Editor

SETTING MEMORY SIZE

Keyware OS-65U maintains a two byte address pointer (Low/High) which points to the current end of memory. These bytes are set by the system when it bootstraps. To modify the workspace size, the new ending address must be POKEd into the locations 132, 133 as in the following:

```
POKE 132,AL:POKE 133,AH:CLEAR
```

where AL and AH are the low and high parts of the address of the new end of the workspace. The 'CLEAR' (or 'RUN') command is required for the change to take effect; the change will remain until the system is rebooted.

The locations given below provide the lower and upper limits to observe when changing memory size; they should not be exceeded.

Low	=	24832
High	=	49152

Figure 7-1 provides a simple map of OS-65U memory showing the major locations in decimal and hexadecimal.

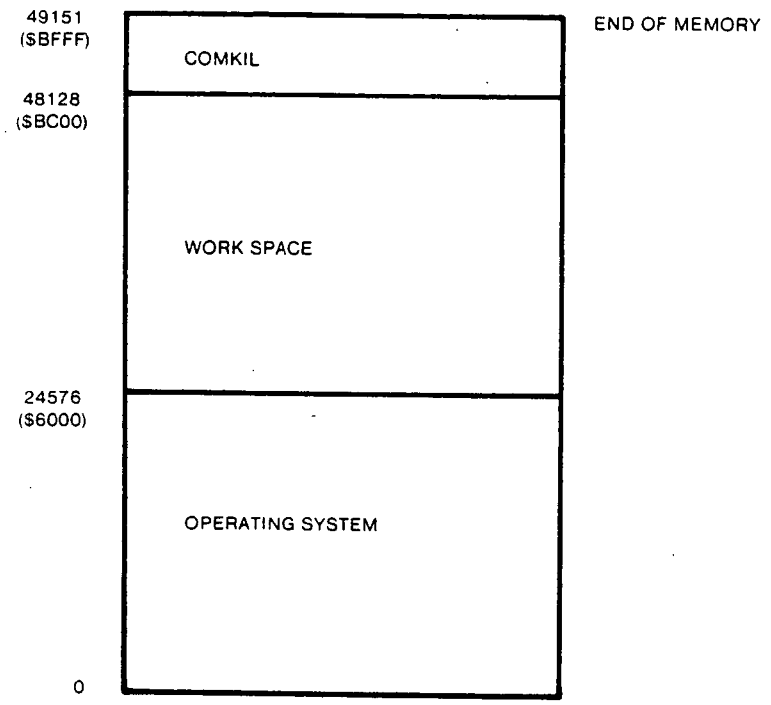

FIGURE 7-1

MERGING BASIC PROGRAMS (FLAGS 13 & 14)

Keyware OS-65U's FLAG 13 & 14 commands provide the user with the capability to merge BASIC programs. This capability makes it possible to insert subroutines or make substantial changes in the program. Essentially, these FLAGs provide a method for bringing a second BASIC program into the work space without erasing one already existing there. The procedure requires the procedures listed below. Assume that a program "TEST2" contains a subroutine between lines 3000 and 3199 that is to be merged into a program called "TEST1".

1) Create a scratch data file (named "SCRAP0") with access rights of R/W (Read/Write).

2) LOAD "TEST2" (Enter LOAD"TEST2")

3) OPEN "SCRAP0" (Enter OPEN"SCRAP0",1)

4) LIST the subroutine into the file (Enter LIST %1,3000 - 3199).

5) PRINT an "OK" to the file (Enter PRINT %1, "OK").

6) CLOSE the file (Enter CLOSE).

7) LOAD "TEST1" (Enter LOAD"TEST1").

8) Enter: FLAG 13

9) Enter: OPEN"SCRAP0",1

10) Enter: INPUT %1,

11) Enter: FLAG 14:CLOSE

12) Now SAVE "TEST1" to disk. The subroutine has now been merged.

This technique simulates a keyboard input to the host program; the error condition (Item 11 above) is created deliberately at that point in order to return control to the keyboard.

Any program lines inserted using this method will be indented one space than they are in the source file being merged into. Therefore, a file that will be merged into another should not use column 71 (the last legal position), because any character in that column will be truncated in the receiving file. The Editor can be used to eliminate the leading space after the merge is complete. The Editor is described in Chapter Six.

THE USR VECTOR

The USR Vector is a pointer to machine code routines. Anywhere a combination of BASIC and machine code occurs, USR provides a 'doorway' between the two. The Transient Functions, for instance, utilize the USR Vector to execute machine code embedded in the programs. The LOAD48 program (discussed next section) utilizes it for loading machine code subroutines into BASIC programs.

The USR Vector consists of memory locations (Low and High bytes) into which is placed the address of the machine code to be executed. These locations in decimal are:

```
8778 = Low byte
8779 = High byte
```

Memory locations 6 and 7 contain pointers to a routine that will pass a 15-bit signed integer from BASIC to machine code. The routine leaves the integer in memory locations B2 (low byte) and B1 (high byte) as outlined below. X represents the value passed from machine code to BASIC, Y represents the value passed from BASIC to machine code.

```
Y = G: X=USR (Y)

In Machine Code

  JSR GETVAL        GETVAL  JMP ($0006)
  LDA $B2 - Low byte
  LDY $B1 - High byte
```

Memory locations 8 and 9 contain pointers that will pass a 15-bit signed integer from machine code to BASIC. The integer is left in the Y register (low byte) and the accumulator (high byte).

```
Y = XX: X = USR (Y)
```

In Machine Code

```
LDY LOWBYT
LDA HIGBYT
JMP ($0008)
```

Assume the following machine code routine is located in front of a BASIC program (see the next section on how to do this).

```
$6000      BASMAL           JSR    GETVAL
                            LDA    $00B2
                    STA     NUMLOW
                    LDA     $00B1
                    STA     NUMHI
                    INC     NUMLOW
                    BNE     MACBAS
                    INC     NUMHI
           MACBAS:  LDY     NUMLOW
                    LDA     NUMHI
                    JMP     ($0008)
           GETVAL:  JMP     ($0006)

           NUMLOW:     .BYTE 00
           NUMHI       .BYTE 00
```

The following is a program for setting up and executing the USR Vector to run the machine code and return to BASIC follows:

```
10 POKE 8778, (00*16+00)
20 POKE 8779, (06*16+00)
30 X=0:Y=6:X=USR(Y)
40 PRINT "Before =";Y;"After =";X
50 END
```

Lines 10 and 20 set up the USR Vector, 30 goes to the machine code and returns, and 40 prints the results.

LOAD MACHINE CODE SUBROUTINES INTO OS-65U BASIC PROGRAM (LOAD48)

LOAD48 provides the programmer with a means of inserting machine code subroutines into OS-65U programs. The assembler in OS-65D or WP1-B may be used to create the machine code. If available, the WP1-B is more useful since it incorporates a full line editor, macro CHANGE and FIND commands, and move and transfer commands.

Before actually moving the machine code into an OS-65U BASIC program, it is important to understand where it will be located in the workspace. The command

NEW"XXXX"

is used to allocate space for the machine code where "XXXX" is the number of bytes to be allocated. The start of the BASIC workspace is offset by the number of bytes specified in the NEW command. When a SAVE command is then executed, all memory transfers to disk start at $6000. Therefore the space allocated by the "NEW 256" command will be SAVEd to disk in front of the BASIC program. When the BASIC program is LOADed, the allocated "dead space" is brought into memory with it. The result is a means of storing machine code in front of a BASIC program so that the machine code "rides" in and out with the program. After that, any BASIC program can use that machine code.

The steps for merging machine code subroutines into BASIC programs are as follows:

1) Enter (Load) the BASIC program.

2) LOAD the BASIC file containing the machine code.

3) Merge the programs and SAVE the resulting program away.

The machine code should be assembled to run at $6000 and up. The assembled machine code must be SAVEd to the WP1-B or 65D diskette. Since Keyware OS-65U cannot read WP1-B or 65D diskettes, the LOAD48 utility is required to call machine code into OS-65U. When RUN, LOAD48 will enter the 65D kernel mode and "A*" is displayed. At this point, the machine code may be "called" into Keyware OS-65U.

The steps below provide the exact sequence for loading a BASIC program to a data file, generating the machine code, and merging the machine code into an OS-65U BASIC program. It is an expansion of the method for merging BASIC programs described in the previous section. The process involves three operations: 1) Loading the BASIC program to a data file, 2) Loading the machine code to OS-65U and 3) Merging the machine code into the BASIC program. To be consistent, this description also uses program "TEST2" which contains a (machine code) subroutine between lines 3000 and 3199 that is to be merged into the program called "TEST1". Press the <Return> key after each step.

Loading the BASIC Program to a Data File

1) Create a scratch data file (name such as "SCRAP0") with access rights of R/W (Read/Write).

2) LOAD "TEST2" (Enter LOAD"TEST2")

3) OPEN "SCRAP0" (Enter OPEN"SCRAP0",1)

4) LIST the subroutine into the file, (Enter LIST 1,3000 - 3199).

5) PRINT an "OK" to the file (Enter PRINT 1,"OK")

6) CLOSE the file (Enter CLOSE).

Loading the Machine Code into OS-65U

1) Enter in immediate mode: RUN"LOAD48","PASS"4

2) Call the machine code into place by entering:
A*CXXXX=YY,Z

   XXXX is the address the machine code is to be called into (normally 6000), YY is the track number and Z is the sector number.

3) 'Warm start' Keyware OS-65U by entering:
A*GXXXX

   XXXX="7E12" for a 32K machine and "BE12" for a 48K machine.

4) Enter: NEW"XXXX"

   XXXX represents the number of bytes to be allocated for the machine code routines plus one. For example, machine code runs from $6000 to $60FF. The statement then would be "NEW 256",(($60FF+1)-$6000)+ $6100 - $6000= $0100+ 256

Merging the Machine Code into the BASIC Program

1) Enter: FLAG 13

2) Enter: OPEN "SCRAP0",1

3) In the immediate mode enter:
INPUT %1,

4) A "SN ERROR" will appear on the screen; enter
FLAG 14 : CLOSE

5) Now SAVE "TEST1" to disk. The subroutine has been merged.

DEBUGGING AIDS

Certain Keyware OS-65U programs and FLAG commands have been designed to provide the programmer with assistance in debugging programs. Specifically, this assistance is for dumping files, printing file contents and tracing programs.

Dump File Contents (FDUMP)

FDUMP provides an ASCII or hexidecimal dump of the file contents on a byte by byte basis. To invoke it, select "Dump File Contents" from the System Utilities menu. The program will prompt through the steps as shown in the following sample dump of program TEST01:

PROMPT	RESPONSE
Name, password?	TEST01,PASS
What device should be searched?	E
The file is nnnnn bytes long.	
File offset, <RETURN> for no offset?	<RETURN> to dump file from beginning; otherwise enter the beginning byte number.
nnnnn bytes can be listed.	
How many bytes to list, <RETURN> for all?	<RETURN> (or number)
Output to what device?	<RETURN> outputs to the console. Other output devices may be specified.
Output format (A)SCII or (H)ex?	A
How many characters per line?	70
(S)ingle or (D)ouble space?	D

The file will then be dumped as specified. Use the Control C key to stop the listing at any point. This will cause the program to prompt

"Continue?"

If 'N' is replied, the program will prompt

Restart File Listing?

If 'Y' is replied to this prompt, the program will return to the "File offset" prompt. If 'N' is replied, the listing will continue.

At ten line intervals, the program will print exclamations (!) as horizontal markers ten characters apart.

ASCII listings will print out non-printable characters as follows:

Carriage returns	#
NULL	-
All others	@

Print File Contents (FPRINT)

The FPRINT program is used for printing out the contents of data files to the user's console or a specified printer. This program allows the programmer to check either sequential or random access data files for errors. Before using FPRINT, it is necesary to understand how OS-65U files are organized. See the chapter on "File Organization and Procedures".

FPRINT is invoked from immediate mode with the command

```
RUN"FPRINT","PASS"
```

Following is a list of prompts along with sample responses for file "TEST01".

PROMPT	RESPONSE
DEVice?	B
Console (C) or printer (P)?	C
Filename?	TEST01
Password?	PASS
Sequential (S) or random (R) type?	R

Sequential Files: If the file were Sequential, the following prompts would be issued. Sample responses follow.

PROMPT	SAMPLE RESPONSE
Starting index?	186368
Finishing index? (0 outputs to end of file)	0

The file is listed from the starting to finishing byte addresses. For each address, the index number is displayed alongside the contents up to, (not including), the carriage return.

Random Access Files: The following prompts are issued for random access files. Sample responses are included.

PROMPT	SAMPLE RESPONSE
Record length?	100
Offset to first record?	50
Starting record number?	5
Finishing record number?	25
Number of fields to be printed?	2
Field 1 location?	0
Field 2 location?	50

Keyware OS-65U random access file organization assumes all records in a file are equal in length. The 'Offset to first record?' prompt requests the the first record location relative to the start of the file. The "Field location" prompts require the location relative to the start of the record.

The contents of each address of the random access file is listed for the specified records.

Tracing Programs

Keyware OS-65U contains a FLAG 7 command which provides the capability to list and execute a program on a line-by-line basis, thereby 'tracing' it through its execution. This can be a useful debugging tool. FLAG 7 is invoked from immediate mode by entering

```
FLAG 7
```

The program is then RUN, using the Control S and Control Q keys to stop and restart the output as required. The program line will appear above the execution line. For instance, the program lines

```
10 INPUT "WHAT IS YOUR NAME";N$
20 INPUT "WHAT IS YOUR ADDRESS";A$
```

would RUN under FLAG 7 as follows:

```
● 10 INPUT "WHAT'S YOUR NAME";N$
   WHAT'S YOUR NAME? (Answer)
● 20 INPUT "WHAT IS YOUR ADDRESS";A$
   WHAT IS YOUR ADDRESS? (Answer)
```

FLAG 7 is turned off by FLAG 8.

SYSTEM MAINTENANCE UTILITIES

Certain Keyware OS-65U utilities are designed to assist the user in properly maintaining/testing the system software and disks. The utilities discussed in this chapter are housekeeping functions used for recovering disk space, applying M/A-COM OSI program patches and configuring the system printer. The programs covered in this chapter should be used by experienced programmers only.

RECOVERING DELETED FILE SPACE (PACKER)

The PACKER utility makes file space reuseable by the operating system; it searches the DIREC* file for entries marked as deleted and closes in the gaps of space they create, thereby "packing" the disk. It then produces a report showing the newly packed disk DIRECtory, bytes used/free, etc.

CAUTION

If a disk error occurs during a system pack, the disk should be presumed destroyed. Always make a backup copy before packing a diskette.

The PACKER program is invoked with the command:

```
RUN"PACKER","PASS"
```

The program issues the warning and prompts that appear below. Sample responses are included with the prompts.

WARNING

Files that do NOT start on sector boundaries or are NOT multiples of a sector in length must NOT be packed (See the DIRECtory program).

PROMPT	RESPONSE
Do you wish to continue?	Y
Password?	PACK
Route DIRECtory listing to (C)onsole or (P)rinter?	C
DEVice to be packed?	B
Are you sure?	Y

The warning is applicable primarily for files created on older versions (prior to version 1.3) of Keyware OS-65U. Check the Sector Boundary (Sec Bnd) column in the DIRECtory listing. The password PACK is required to access the PACKER program.

The program then lists the newly packed DIREC* file and displays current disk space information. Press <RETURN> to exit to immediate mode.

MODIFYING DISK CONTENTS (CHANGE)

The CHANGE program enables the user to modify the contents of the disk on a byte-to-byte basis in hexadecimal. The primary use of the program is to enter M/A-COM OSI patches to Keyware OS-65U. Since specific instructions are included with the patches, the descriptions contained here are brief. The CHANGE program is invoked from immediate mode with the command:

```
RUN"CHANGE","PASS"
```

The program issues the prompts as listed and explained below.

```
MODE: HEX(H), DEC(D)?
ADDRESS OFFSET?
ADDRESS?
```

MODE:	File format, (H)ex or (D)ecimal. M/A-COM OSI patch information will specify this.
ADDRESS OFFSET:	This amount will be added to the disk address in order to synchronize it with the memory address. Generally, the response to this prompt will be C00. In a few cases, patches will be made in Track 0 which will require a response of -2AFD (floppy based diskette) or -2B00 (hard disk based diskette). This information will be supplied with M/A-COM OSI patches.
ADDRESS:	Address at which to make first change. The address will be supplied with M/A-COM OSI patches.

The program then lists the file addresses sequentially in hexadecimal. Printable ASCII characters are also displayed.

Several editing functions are supplied with the CHANGE program to control the program and apply patches. The function keys follow:

/ (slash)	Moves to next higher address.
\ (backslash)	Moves to next lower address.
. (period)	Returns to "ADDRESS?" prompt. May enter new values, if necessary.
. (second period)	Returns to "ADDRESS OFFSET?" prompt. May enter new values, if necessary.
X	Ends program and updates disk by writing relevant section. Returns user to immediate mode.

Following is a sample use of the CHANGE program's editing functions:

```
PROMPT                                  RESPONSE

MODE: HEX(H),  DEC(D) ?                  H
DEVice ?                                 B
ADDRESS OFFSET ?                         C00
ADDRESS ?                                020
00000555          20    ?                /
00000556      s   73    ?                /
00000557          0A    ?                /
00000558          A2    ?                /
00000559          00    ?                /
0000055A          20    ?                .
ADDRESS     ?                            .
ADDRESS OFFSET       ?                   0
ADDRESS  ?                               6200
00006200      D   44    ?                /
00006201      I   49    ?                /
00006202      R   52    ?                /
00006203      E   45    ?                /
00006204      C   43    ?                /
00006205      *   2A    ?                /
00006206          FF    ?                \
00006205      *   2A    ?                \
00006204      C   43    ?                \
00006203      E   45    ?                \
00006202      R   52    ?                \
00006201      I   40    ?                \
00006200      D   44    ?                \
000061FF          0D    ?                X

OK
```

DEVICES 3 & 8 BASE ADDRESS PORT SET (PRTSET)

The PRTSET program is used to configure the operating system to hardware environments; it provides the capability of configuring physical DEVices 3 & 8 drivers for any serial port address. PRTSET replaces the requirement of using POKEs to change the ports; this procedure was required with earlier versions of OS-65U. POKEing the port offsets is still an available procedure, but PRTSET eliminates the necessity for it.

As programmed by M/A-COM OSI, DEVice driver 3 will access address $CF02 and DEVice 8 will access $CF00. To change them, the PRTSET program requests the DEVice 3 & 8 addresses (high and low bytes). Figure 8-1 shows the relationship of the physical ports to the hexadecimal addresses.

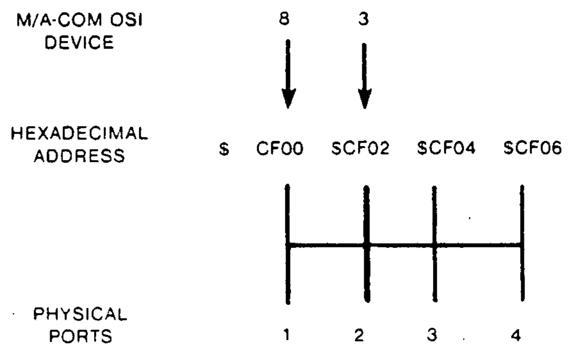

FIGURE 8-1

The PRTSET program may be used with floppy diskettes only. Hard disk system users should make the PRTSET changes on their hard disk based floppy diskette and then copy the entire diskette contents to the hard disk. The PRTSET program is involed from immediate mode with the command

RUN"PRTSET"

and issues the following series of prompts:

```
Please make sure you have read the manual before using this utility!!!

Do you wish to continue?

DEVice?

Enter high byte of DEVice 3 base address port (0-FF)?
```

Enter low byte of DEVice 3 base address port (00-0F)?

Enter high byte of DEVice 8 base address port (0-FF)?

Enter low byte of DEVice 8 base address port (00-0F)?

Please wait

You must reboot to make the changes effective.

Enter a <cr> to continue?

CAUTION

It is very important that a copy of the system diskette has been made before using PRTSET. PRTSET modifies the operating system; if a Read or Write error occurs during execution, the state of the operating system is unknown and should be viewed as defective.

Example To change DEVice 3 to a serialport located at address $CF04 and DEVice 8 to a serial port located at address $CF06 requires the following series of instructions to PRTSET:

PROMPT	RESPONSE
Please make sure you have read the manual before using this utility!!!	
Do you wish to continue?	Y
DEVice?	A
Enter high byte of DEVice 3 base address port (0-FF)?	CF
Enter low byte of DEVice 3 base address port (00-FF)?	04
Enter high byte of DEVice 8 base address port (0-FF)?	CF
Enter low byte of DEVice 8 base address port (00-0F)?	06
(Please Wait)	
You must reboot to make changes effective.	
Enter a <cr> to continue?	<cr>

As indicated, the changes are recorded, but do not become effective until the system is rebooted.

CONFIGURING SYSTEM PRINTER (PRTMAP)

This program maps logical print device #5 to physical print device #3, 5, 6 or 8 (See "Displaying and Printing File Contents" in an earlier chapter of this manual for definition of these devices). The #5 logical print device is the only one with print/paging control; it outputs to a Centronics-type parallel port. The PRTMAP program allows the output of the #5 logical device to be routed to other physical printers. In practice, the user may specify #5 as the printer; the system will route the output with pagination to whatever device the user has specified with PRTMAP. It is necessary to use this procedure each time the user logs in or reboots the system.

To use PRTMAP, enter the command

```
RUN"PRTMAP","PASS"
```

from immediate mode. Program prompts follow with sample responses:

PROMPT	SAMPLE RESPONSE
Select the physical driver device for the system printer (3, 5, 6, or 8 only)?	8
Do you wish auto paging enabled (Y/N)?	Y
Enter the page length in inches	11
Enter the number of lines per inch the printer is set for (Press a <cr> for a default of 6 lines/inch)	<cr>
There are a maximum of 66 lines per page	
Enter the total number of blank margin lines required (Remember both the top and bottom margins) (Press a <cr> for a default of 6)	<cr>
Margin lines/page? Set the printer paper to the location of where you want the first line to print. Press a <cr> after doing this?	10
Does the system printer accept CHR$(12) for top of form? (Y/N)?	Y

NOTE: If number 6 (word processing printer) is selected as the physical driver device, the program will prompt "Word Processing Driver (WPDRIV) must be enabled". See the next section in this chapter for the procedure to do this.

The last prompt asks if the printer will accept standard ASCII (12) as a top of form command.

PARALLEL WORD PROCESSING DRIVER (WPDRIV)

This program configures DEVice #6 as the output device for the Keyware OS-65U word processing printer. If paging is required, it can be enabled by the use of PRTMAP as described in an earlier section of this chapter. Note that enabling WPDRIV disables the print spooling capability. To enable WPDRIV, enter:

```
RUN"WPDRIV","PASS"
```

from immediate mode. Device #6 will be designated as the output device automatically. To disable it, enter the RUN command as above; the program will prompt

```
(E)nable or (D)isable?
```

Enter "D".

Should the printer become stalled for any reason, the program will prompt

```
Printer failed, try again (Y/N)?
```

If "N" is replied, the prompt

```
Printer off-line
```

printer. If the printer is operational, the prompt

```
WPDRIV is operational
```

will appear. If the printer is not operational, the prompt

```
Printer failed
```

appears. To disable the printer driver, enter

```
RUN"WPDRIV","PASS"
```

again from immediate mode. This will produce the prompt

```
(E)nable or (D)isable WPDRIV?
```

Entering 'D' will disable WPDRIV.

THE OS-65U DISK SYSTEMS MANAGER

This chapter discusses:

- What is the OS-65U Disk Systems Manager?
- Software required by the Disk Systems Manager.
- Disk Structure Maps.
- Converting from SYSDIR to the Disk Systems Manager.
- How to install the Disk Systems Manager Software.
- How to configure a hard disk with the Disk Systems Manager.
- How to modify the configuration of a hard disk.
- How to use a Masterkey computer on which the Disk Systems Manager has been installed.

For brevity, the following sections often refer to the Disk Systems Manager by its acronym, DSM.

WHAT IS THE DISK SYSTEMS MANAGER

The Disk Systems Manager (DSM) is a utility for subdividing a Masterkey computer's hard disk(s) into multiple regions, each of which can then be accessed independently of the rest. Two levels of subdivision are possible with the Disk Systems Manager:

1. A hard disk may be divided into regions each of which contains its own operating systems. As of OS-65U Version 1.44, the operating systems available are OS-65U and CP/M Version 2.25.

2. The region allocated to OS-65U can be further subdivided into file systems. Each file system has its own set of files, and can be accessed independently of all other file systems.

The process of using the DSM to subdivide a hard disk into areas for different operating systems, and then subdividing the area of the OS-65U operating system into sub-areas for different file systems, is termed configuring the hard disk.

The first operating system on a hard disk configured with the Disk Systems Manager must be OS-65U.

Every OS-65U file system must contain copies of all system and user files that will be needed by any user when connected to that file system. File System 0 should contain a copy of every OS-65U system file, so that it serves as a central repository for OS-65U. File System 0 is sometimes called the Base System.

When a Masterkey computer has two hard disks (Device E and Device F), the DSM can be used on either disk or on both. The two disks remain separate realms; use of the DSM on either disk has no effect at all on the other disk. For simplicity, this chapter discusses only Device E. All the same techniques can be used on Device F by making it the current device before applying the techniques.

Accessing a Configured Hard Disk

When the DSM has been used to configure a hard disk, a user who wishes to access that disk performs the following operations:

1. Select that disk as the current device.

2. Select the desired operating system.

3. Select the desired file system (for OS-65U)

DISK SYSTEMS MANAGER SOFTWARE

The Disk Systems Manager requires the following software:

1. OS-65U Version 1.44 or later.

2. The files SYSTEM, SYSDIR, SYSUTL, SYSCR*, SYSREN, SYSDEL, SYSDR*, and SYSTR*.

3. The files BEXEC* and SYSDIR in every OS-65U file system.

OS-65U (including BEXEC*) is on the OS-65U System Diskette(s). The other files mentioned are on the Disk Systems Manager Diskette. Be sure the version numbers on the diskettes match, or the system will not work correctly.

CAUTION

Do not confuse the SYSDIR program that is part of the DSM with the SYSDIR program that is a standard part of OS-65U. The standard SYSDIR is a simple program which provides some of the capabilities of the DSM. When the DSM is used, its SYSDIR replaces the old SYSDIR in all OS-65U file systems.

DISK STRUCTURE MAPS

Installing the Disk Systems Manager involves three basic stages:

1. Make a complete map of the structure that the configured hard disk is to have.

2. Install the Disk Systems Manager software on the hard disk.

3. Use the Disk Structure Map and the Disk Systems Manager to configure the disk.

Of these stages, making a Disk Structure Map is the most critical. The needed map specifies a name, password, and size for every region on the disk. Once this map has been established, configuring the disk is largely a matter of entering information from the map in response to various prompts.

Making a Disk Structure Map

Making a Disk Structure map is just a matter of assigning values to all the variables applicable to the described disk structure. All that matters is that the map gives a value to every variable, and that all size values are consistant with each other and with the maximum capacity of the hard disk.

The following paragraphs give details on how to select values for Disk Structure Map variables.

Selecting Names: The name of an operating system or file system may contain up to 24 ASCII characters. Blanks may be used. Every operating system name must be unique among the operating system names, and every file system name must be unique among the file system names.

If the string "(HD)" appears anywhere in a system name, that system will be accessible only by a user in single-user mode. If the string "(NET)" appears anywhere in a system name, that system will be accessible only by a workstation.

Selecting Passwords: A password may contain up to 8 alphanumeric characters. Blanks may be used.

Selecting the Size of File System 0: Three points must be kept in mind:

1. File System 0 (the Base System) must be at least 1,000,000 bytes long. This length provides enough space for all OS-65U standard, Timesharing, Print Spooling and Networking files, and for the Disk Systems Manager files.

2. The Base System must be defined so that it does not truncate any file already on the hard disk. To guard against this possibility, get a directory (#1 on the Main Menu) of Device E, add the start address and length of the last file shown, then add 100,000, which is the space that will be used by the Disk Systems Manager files when they are installed. File System 0 must be larger than the sum obtained.

3. At least 250,000 unused bytes should be provided in File System 0 to allow for expansion.

Selecting the Sizes for Other File Systems: The minimum size for a file system other than File System 0 is 25088 bytes for the system area. Other than that, no absolute rules apply. The purpose of each file system, must be considered, and space requirements projected accordingly.

Selecting the Size of the OS-65U System Area: The only rule is that there must be enough room in the OS-65U area to hold all desired file systems. The minimum is therefore 1,000,000 bytes, the size of File System 0.

Minimum Size for CP/M: If a CP/M operating system area is created, it must contain at least 300,000 bytes to hold CP/M itself, plus more space for user files. A size of less than 1,000,000 bytes would rarely be useful in practice.

CONVERTING FROM SYSDIR TO DSM

The SYSDIR program is a simplified predecessor of the Disk Systems Manager. If your hard disk has not been configured with SYSDIR, you can skip to the next section.

If your hard disk has been configured with SYSDIR and you want to switch to DSM, you have two choices:

1. Recreate with DSM the same structure that had previously been created with SYSDIR.

2. Create with DSM a structure different from the one previously created with SYSDIR.

Recreating the Existing Structure

This is the easiest method. It can be used if:

1. The SYSDIR File System 0 is at least 1,000,000 bytes long.

2. There are at least 100,000 bytes free in File System 0, to hold the Disk Systems Manager files.

The steps for converting from SYSDIR to DSM are:

1. From the Main Menu select #6, "Select File System".

2. A menu listing the name, start address, and length of every hard disk region will be displayed. Use this information to fill out the Disk Structure Map.

3. Install DSM as described in the following sections. The fact that the Disk Structure Map used defines the same structure that was defined by SYSDIR is all that is needed to create the SYSDIR structure under DSM.

Creating a New Structure

The steps are:

1. Back up all file systems onto floppy disks, as described in Chapter 2. Use the Volume Copier for file systems too large to fit on one floppy.

2. Initialize the entire hard disk as described in Chapter 15.

3. Install OS-65U as described in Chapter 15.

4. Install DSM as described in the following sections, using a Disk Structure Map that defines the desired new structure.

5. Copy files as described from the floppy disks to file systems in the new disk structure.

WARNING

The Disk Systems Manager contains a file named SYSDIR, which replaces the stand-alone SYSDIR in every File System when the DSM is used. When files backed up in Step 1 above are restored in Step 5, the old SYSDIR could overwrite the new SYSDIR that was installed in Step 4, thereby destroying an essential part of the Disk Systems Manager.

To prevent this problem, use INSTAL to restore the diskettes containing backed up files to each file system. To the INSTAL prompt:

Do you wish to copy all files on the source device?

answer "N". INSTAL will then display all source device file names one at a time, and ask whether you do or do not want each one to be transferred. When INSTAL asks about SYSDIR, reply "N". This will prevent the old SYSDIR from being copied, thereby protecting the new one.

INSTALLING THE DSM SOFTWARE

The following are steps for installing the Disk Systems Manager software on a hard disk. Installing the software does not per se configure the hard disk; it only makes disk configuration possible. These steps assume that the hard disk has been completely initialized, and that the standard OS-65U software has been installed on it, as described in Chapter 15.

1. Boot the computer from the hard disk.

2. If the disk is currently configured with SYSDIR, select from the Main Menu #6, "Select File System". Choose File System 0, then return to the Main Menu.

3. Use the RENAME utility as described in Chapter 2 to change the name of the file SYSDIR to ZYSDIR. This preserves a copy of the old SYSDIR and any information coded into it.

4. Enter Immediate Mode, and use the INSTAL program as described in Chapter 15 to copy all files on the Disk Systems Manager Diskette to the hard disk.

The Disk Systems Manager software has now been installed. The next step is to use it to configure the hard disk.

CONFIGURING A HARD DISK

The following are the steps for configuring a hard disk. Before you perform them, be sure you understand everything discussed in this chapter so far, and have made a Disk Structure Map and installed the Disk Systems Manager software as described above.

The general stages of hard disk configuration are:

1. Delete the existing OS-65U operating system that is built into the Disk Systems Manager. (No files are deleted by this step.)

2. Create OS-65U File System 0.

3. Create the OS-65U Operating System Area.

4. Create the CP/M Operating System Area (optional).

5. Create additional OS-65U File Systems (optional).

Note

The following steps assume a CD-7 hard disk, with a size of 7,340,032 bytes and a track length of 28,762 bytes. Your hard disk may have a different size and track length, but the difference does not alter the DSM installation sequence in any way. It will affect only the numbers that appear in some DSM displays.

Delete the Existing OS-65U Operating System

1A. Boot the computer from the hard disk.

2A. The Operating Systems Directory will be displayed, as follows:

```
                    Operating Systems

No.              Name                  Address     Length

0>     Operating System Utilities         -           -

1>     OS-65U System                      0        7340032
```

3A. From the Operating Systems Directory select #0, "Operating System Utilities". The DSM will request a password, which is PASS.

4A. The Operating System Utilities Menu will be displayed, as follows:

```
1> Create an operating system
2> Rename an operating system
3> Delete an operating system
4> Print an operating system directory
5> Select a new device

Enter a <Return> to select an operating system.

Your selection?
```

Select #3, "Delete an operating system".

5A. The DSM prints:

```
There are 1 systems in the directory.

What is the number of the system to be deleted?
```

Reply by typing in a "1".

6A. The DSM will request a password. Type in "PASS".

7A. The DSM will print the Deletion Verification Display. A typical example is:

```
Delete Operating System Utility

There are 1 systems in the directory

Delete system # 1?

System name        OS-65U System
Password           PASS
Location           0
Size               7340032 bytes

Are you sure?
```

8A. The DSM will print:

```
System number 1 has now been deleted.

Enter a <Return> to continue.
```

Enter a <cr>. The Operating System Utilities Menu (see Step 4A) will be redisplayed.

CAUTION

Do not leave the Operating System Utilities Menu at this point. If you do, the DSM will automatically re-create the OS-65U system just deleted, and you will have to start over.

Create OS-65U File System 0

1B. From the Operating System Utilities Menu, select #1, "Create an Operating System".

2B. The DSM prints:

```
What is the size of the Base System
<1000000 bytes>?
```

Reply by entering the value for 65U-FS0-SIZE from the Disk Structure Map.

3B. The DSM echoes the size you entered, rounded up to the nearest whole disk track, and asks for verification. If you had entered 1500000 in Step 10, the system would print:

```
Base system size of 1519616 <Y/N>?
```

Answer "N" to get another chance to set the size of File System 0. If the size as shown is acceptable, answer "Y".

4B. Once the File System 0 size has been given, the DSM requests the File System 0 password:

```
What is the eight character password for the base system?
```

Enter 65U-FS0-PASS.

File System 0 has now been created. Its name, which is established by the DSM and cannot be changed, is "Base System".

Create the OS-65U Operating System Area

1C. The DSM prints:

```
There are 0 systems in the directory.

7340032 bytes free after System 0.

What is the name of the new system?
```

Enter 65U-SYSTEM-NAME>

2C. The DSM prints:

```
What is the length of the system in bytes?
```

Enter 65U-SYSTEM-PASS.

3C. The DSM prints:

```
What is the length of the system in bytes?
```

Enter 65U-SYSTEM-SIZE. The system will round that size up to the nearest whole disk track.

4C. The DSM will echo all the values entered, plus the OS-65U area start address, the system would print:

```
Create as system # 1

System name        OS-65U
Password           PASS
Location           0
Size               2523136 Bytes

Is this alright?
```

5C. If any of the values shown is unacceptable, enter "N". The DSM will print:

```
Do you want to create another system?
```

Answer "Y", then return to Step 1C and start over. The values for 65U-FS0-SIZE and 65U-FS0-SIZE and 65U-FS0-PASS are not affected by such a re-try.

6C. If all the values shown for the OS-65U system are satisfactory, answer "Y" to the question "Is this alright?". The DSM will print:

```
65U-SYSTEM-NAME created as system number 1
```

where 65U-SYSTEM-NAME is the name entered in Step 1C.

7C. The DSM prints:

```
Do you want to create another system?
```

If you want to create a CP/M system, respond "Y", then go to step 1D, below under "Create the CP/M Operating System Area".

8C. If you don't want to create a CP/M area, respond "N". The DSM will print:

```
Enter a <Return> to continue?
```

Enter <cr>. The Operating System Utilities Menu will be displayed. Enter another <cr>. The Operating Systems Directory will be displayed.

9C. Go to Step 1E below under "Create Additional OS-65U File Systems".

Create the CP/M Operating System Area

1D. The DSM prints:

```
There are 1 systems in the directory

NNNNNNN bytes free after System 1

What is the name of the new system?
```

where NNNNNNN is the difference between HARD-DISK-SIZE and 65U-SYSTEM-SIZE

Enter CP/M-SYSTEM-NAME.

2D. The DSM prints:

```
What is the password of this system?
```

Enter CP/M-SYSTEM-PASS.

3D. The DSM prints:

```
What is the length of the system in bytes?
```

Enter CP/M-SYSTEM-SIZE. The DSM will round that size up to the nearest whole disk track.

4D. The DSM will echo all the values entered, plus the CP/M area start address, and would print:

```
Create as system # 2

System name        CP/M
Password           OPSYS
Location           2523136
Size               414080 bytes

Is this alright?
```

5D. If any of the values shown is unacceptable, enter "N". The DSM will print:

```
Do you want to create another system?
```

Answer "Y", then return to Step 1D and start over. The values for the already created OS-65U system are not affected by such a re-try.

6D. If all the values shown for the CP/M file system are satisfactory, answer "Y" to the question "Is this alright?". The DSM will print:

```
CP/M-SYSTEM-NAME created as system number 2
```

where CP/M-SYSTEM-NAME is the name entered in step 1D.

7D. The DSM then prints:

```
Initialize the new system?
```

If you are recreating a CP/M System previously created with SYSDIR, enter "N". Otherwise, answer "Y".

8D. The system will print:

```
Do you want to create another system?
```

You may create another CP/M system, if there is room for it, by answering "Y", going back to step 1D, and going through Steps 1D-8D again with an appropriate CP/M-SYSTEM-NAME, CP/M-SYSTEM-PASS, and CP/M-SYSTEM-SIZE. Any number of CP/M systems can be created in this way.

9D. If you don't want to create any more CP/M systems, respond "N". The DSM will print:

```
Enter a <Return> to continue?
```

Enter <cr>. The Operating System Utilities Menu will be displayed. Enter another <cr>. The Operating Systems Directory will be displayed.

10D. If you want to create OS-65U file systems in addition to File Systems 0, go to Step 1E in the next section. Otherwise, continue to Step 11D.

11D. Installation of the Disk Systems Manager is now complete. You are at the highest level of the OS-65U Menu Tree. Be sure you read "Using a Masterkey Computer with the Disk Systems Manager" before you try to do anything more with the computer.

Create Additional OS-65U File Systems

Any number of file systems can be created using the following steps, by running through them once for each file system. The first file system must be File System 1, followed by Systems 2, 3, etc. File System 0 already exists, having been created in Steps 1B-4B.

Because of the iterative nature of file system creation, the following steps are represented as a loop. The loop control variable is n. Initially, n is set to 1, so that the first iteration a reference to, say, 65U-FSn-NAME is a reference to 65U-FS1-NAME. The second time through, n=2, and the reference is to 65U-FS2-NAME, etc.

1E. Assign to n the value 1

2E. From the Operating Systems Directory, select #1, 65U-SYSTEM-NAME. ("65U-SYSTEM-NAME" stands for the name entered previously in Step 1C. That name is what actually appears in the Operating Systems Directory.)

3E. The DSM will request a password. Enter 65U-SYSTEM-PASS.

4E. The File Systems Directory will be displayed, as follows:

```
                    OS-65U FILE SYSTEMS

No.   Name                                Address     Length

0     OS-65U File Systems Utilities          -           -

System number?
```

Enter a "0".

5E. The DSM requests a password. Enter 65U-FS0-PASS.

6E. The File System Utilities Menu will be displayed, as follows:

```
1> Create a file system
2> Rename a file system
3> Delete a file system
4> Print/Display directories
5> Select a new device
```

```
 6> Transfer programs/files
 7> OS-65U utilities menu
 8> Transient utilities menu
 9> Multi-User menu
10> Unlock system

Enter a <Return> to select a file system.

Your selection?
```

Select #1, "Create a File System".

The DSM prints:

```
There are XX systems in the directory

There are NN bytes free after system XX

What is the name of the new system?
```

where XX is the number of existing file systems other than File System 0, and NN is the number of bytes that have been allocated to the OS-65U operating system area but have not been allocated to a file system.

Enter 65U-FSn-NAME.

8E. The DSM prints:

```
What is the password of this system?
```

Enter 65U-FSn-PASS.

9E. The DSM prints:

```
What is the length of the system in bytes?
```

Enter 65U-FSn-SIZE. The size will be rounded up to the nearest whole disk track.

10E. The systm will echo all the values entered, plus the 65U-FSn area start address and the DSM would print:

```
Create as system # 1

System name        MORASS
Password           DISASTER
Location           1519616
Size               774144 bytes
```

11E. If any of the values shown is unacceptable, enter "N". The DSM will print:

Do you want to create another system?

Answer "Y", then return to Step 7 and start over. No values already established will be affected by such a re-try.

12E. If all the values shown for the file system are satisfactory, answer "Y" to the question "Is this alright?". The system will print:

65U-FSn-NAME created as system number n

Initialize the new system?

13E. If you respond "Y", the file system will be initialized. Any files it contains will be destroyed, as with all OS-65U disk initializations. If you respond "N", no initialization will take place.

If you are recreating a disk structure originally created with SYSDIR, or if you are sure that the disk area comprising the file system has already been initialized, respond "N". Otherwise, respond "Y". If you respond "Y", the DSM will print:

Are you sure?

Reply "Y" to start the initialization, "N" to abort it.

14E. Whether or not you initialize the file system, the DSM will inspect it to be sure it has a copy of DIREC*. If it does not find DIREC*, it prints:

** NO DIRECTORY **

Do you want to create a directory file <No>?

15E. The DSM prints:

How many directory entries are needed <224>?

Enter a number greater than the number of files that will ever exist in the file system. There is no purpose to creating less than 224 directory entries: no space will be saved, and the inconvenience of running out of directory entries at a critical time can be enormous.

The DSM will take about 1 minute to set up the DIREC* file.

16E. In every case, the DSM now copies BEXEC* and SYSDIR from File System 0 to the file system being created. These files are essential in every file system. The DSM prints various messages detailing the progress of the copy operations.

17E. The DSM prints:

Do you want to create another system?

If you do want to create another file system, add 1 to the current value of n, enter "Y", then go to Step 7E.

18E. If you do not want to create any more file systems, enter "N". The DSM will print:

```
Enter a <Return> to continue?
```

Type <cr>. The File System Utilities Menu will be displayed. Type another <cr>. The File Systems Directory will be displayed. Type a "/". The Operating Systems Directory will be displayed.

19E. Installation of the Disk Systems Manager is now complete.

You are now at the highest level of the OS-65U/DSM Menu Tree. Be sure you read "Using a Masterkey Computer with the Disk Systems Manager" before you try to do anything more with the computer.

For information on how to copy files to a file system, see below under "The File System Utilities Menu".

MODIFYING A HARD DISK CONFIGURATION

There are basically two levels of hard disk reconfiguration:

- Modifying Operating Systems
- Modifying OS-65U File Systems

These two levels are similar in many ways, but they differ in various details. Each is described separately in the following two sections.

MODIFYING OPERATING SYSTEMS

This section assumes that you are familiar with everything described in this chapter so far.

Modification of operating systems is done via the Operating System Utilities Menu. That menu looks as follows:

```
1> Create an operating system
2> Rename an operating system
3> Delete an operating system
4> Print an operating system directory
5> Select a new device

Enter a <Return> to select an operating system.

Your selection?
```

Accessing the Operating System Utilities Menu

1. Boot the computer from the hard disk.
2. The Operating Systems Directory will be displayed. Select #0, "Operating System Utilities".
3. The DSM will request a password. Enter 65U-FS0-PASS.
4. The Operating System Utilities Menu will be displayed.

Create an Operating System

Creating an operating system at any time requires the same information and follows the same steps that would be needed to create that system when the DSM is first installed. The method is:

1. Make a Disk Structure Map that includes the desired new operating system. If there is any free space in the current structure due to an operating system having been deleted, the DSM will put the new operating system there if it will fit. The Disk Structure Map should be made out accordingly.
2. For a new OS-65U operating system, follow the steps given previously under the headings "Delete the Existing OS-65U Operating System", "Create OS-65U File System 0", and "Create the OS-65U Operating System Area". Use the current 65U-SYSTEM-PASS in Step 6A, rather than "PASS". Stop at Step 8C.
3. For a new CP/M operating system, access the Operating System Utilities Menu and select #1, "Create an Operating System". Then follow the steps given previously under "Create the CP/M Operating System Area". Stop at Step 9D.

Rename an Operating System

1. Note the number of the operating system you want to rename. This number appears in the Operating Systems Directory that is displayed when the computer is booted from the hard disk.
2. Access the Operating System Utilities Menu and select #2, "Rename an Operating System".
3. The DSM requests the number of the system to be renamed. Enter the number obtained in Step 1.
4. The DSM requests the password of the system to be renamed. Enter that password.

5. The DSM displays the name, password, start address, and size

   Is this the correct system?

   If it is not the correct system, answer "N" and go to Step 3. If the system is the correct one, answer "Y".

6. The DSM prints:

   What is the new name for this system?

   Enter the desired name.

7. The DSM echoes the new name, then prints:

   What is the system's new password?

   Enter the desired password.

8. The DSM displays the new name, new password, start address, and size of the system, then asks:

   Is this correct?

   If it is not correct, answer "N" and go to Step 6.

9. If the new name and password are correct, enter "Y". The DSM will print:

   System number n has been renamed.

   Enter a <Return> to continue.

   where n is the number entered in Step 3.

10. Enter a <cr>. The Operating System Utilities Menu will be displayed. Enter another <cr>. The Operating Systems Directory will be displayed.

Delete an Operating System

1. Note the number of the operating system you want to delete. This number appears in the Operating Systems Directory that is displayed when the computer is booted from the hard disk.

2. Select #3, "Delete an Operating System", from the Operating System Utilities Menu.

3. The DSM requests the number of the system to be deleted. Enter the number obtained in Step 1.

4. The DSM requests the password of the system to be deleted. Enter that password.

5. The DSM displays the name, password, start address, and size of the system, then asks:

   Are you sure?

   If you do not want to delete the system, answer "N", then go to Step 7.

6. If you want to delete the system shown, enter a "Y". The DSM will print:

   System number n has now been deleted.

   Enter a <Return> to continue.

   where n is the number entered in Step 3.

7. Enter a <cr>. The Operating System Utilities Menu will be displayed. Enter another <cr>. The Operating Systems Directory will be displayed.

WARNING

If you delete OS-65U you should create a new OS-65U before you leave the Operating System Utilities Menu. If you do not, the DSM will create an OS-65U that occupies the entire hard disk, resulting in the deletion of all other operating systems and OS-65U file systems.

Other Operating System Utilities Menu Selections

For information on Selections #4 and #5 from the Operating System Utilities Menu, see below under "The Operating System Utilities Menu".

MODIFYING OS-65U FILE SYSTEMS

This section assumes that you are familiar with everything in this chapter so far, except possibly the previous section.

Modification of OS-65U file systems is done via the File System Utilities Menu. That menu appears as follows:

```
1> Create a file system
2> Rename a file system
3> Delete a file system
4> Print/Display directories
5> Select a new device
6> Transfer program/files
```

```
 7> OS-65U utilities menu
 8> Transient utilities menu
 9> Multi-User menu
10> Unlock system

Enter a <Return> to select a file system

Your selection?
```

Accessing the File System Utilities Menu

1. Boot the computer from the hard disk.
2. The Operating Systems Directory will be displayed. Select #1, 65U-SYSTEM-NAME.
3. The DSM will request a password. Enter 65U-FS0-PASS.
4. The File Systems Directory will be displayed. Select #0, "File Systems Utilities".
5. The DSM will request password. Enter 65U-FS0-PASS.
6. The File System Utilities Menu will be displayed.

Create a File System

Creating a file system at any time requires the same information and follows the same steps that would be needed to create that system when the DSM is first installed. The method is:

1. Make a Disk Structure Map that includes the desired new file system. If there is any free space in the current structure due to a file system having been deleted, the DSM will put the new file system there if it will fit. The Disk Structure Map should be made out accordingly.
2. Select #1, "Create a File System" from the File System Utilities Menu.
3. Follow Steps 7E and 18E above under "Create Additional OS-65U File Systems".

Rename a File System

1. Note the number of the file system you want to rename. This number appears on the File Systems Directory that is displayed in Step 4 under "Accessing the File System Utilities Menu".
2. Access the File System Utilities Menu and select #2, "Rename a file system".

3. The DSM requests the number of the system to be renamed. Enter the number obtained in Step 1.

4. The DSM requests the password of the system to be renamed. Enter that password.

5. If the system selected in Step 3 was File System 0, the DSM prints:

   What is the new password for the base system?

   Enter the desired password. The system will echo the password then print:

   Enter a <Return> to continue?

   Go to Step 11.

6. The DSM displays the name, password, start address, and size of the system, then asks:

   Is this the correct system?

   If it is not the correct system, answer "N" and go to Step 3. If the system is the correct one, answer "Y".

7. The DSM prints:

   What is the new name for this system?

   Enter the desired name.

8. The DSM echoes the new name, then prints:

   What is the system's new password?

   Enter the desired password.

9. The DSM displays the new name, new password, start address, and size of the system, then asks:

   Is this correct?

   If it is not correct, answer "N" and go to Step 7.

10. If the new name and password are correct, enter "Y". The DSM will print:

    System number n has been renamed.

    Enter a <Return> to continue.

    where n is the number entered in Step 3.

11. Enter a <cr>. The File System Utilities Menu will be displayed. Enter another <cr>. The File Systems Directory will be displayed. Enter a "/". The Operating Systems Directory will be displayed.

Delete a File System

1. Note the number of the file system you want to delete. This number appears in the File System Directory that is displayed in Step 4 above under "Accessing the File System Utilities Menu".

2. Access the File System Utilities Menu and select #3, "Delete a File System".

3. The DSM requests the number of the system to be deleted. Enter the number obtained in Step 1.

4. The DSM requests the password of the system to be deleted. Enter that password.

5. The DSM displays the name, password, start address, and size of the system, then asks:

   Are you sure?

   If you do not want to delete the system, answer "N", then go to Step 7.

6. If you do want to delete the system, answer "Y", then go to Step 7.

   System number n has now been deleted.

   Enter a <Return> to continue.

   where n is the number entered in Step 3.

7. Enter a <cr>. The File System Utilities Menu will be displayed. Enter another <cr>. The File Systems Directory will be displayed. Enter a "/". The Operating Systems Directory will be displayed.

USING A MASTERKEY COMPUTER WITH THE DISK SYSTEMS MANAGER

The rest of this chapter is written for the user who needs to know how to use a Masterkey computer on which the Disk Systems Manager has been installed.

Before reading the rest of this chapter, be sure you have read the section "What is the Disk Systems Manager" at the beginning of the chapter. That is the only section you need to have read before reading the following sections.

The rest of this chapter takes you through the various menus and directions that you may encounter when using a Masterkey computer with the DSM. It tells you what each selection from each menu does. For selections of interest only to those who want to configure or reconfigure a hard disk, references to the appropriate preceding section(s) are given. For the remaining selections complete information is presented.

The following sections also discuss the effect of the DSM on Timesharing and Network users.

THE OPERATING SYSTEMS DIRECTORY

A typical Operating Systems Directory is as follows:

Operating Systems

No.	Name	Address	Length
0>	Operating System Utilities	-	-
1>	OS-65U	0	2523136
2>	CP/M	2523136	4014080

The Operating Systems Directory is the top level of the OS-65U/DSM menu tree. This directory is displayed in the following five instances:

1. When a stand-alone computer that uses DSM is booted from the hard disk.

2. When a network user accesses the network, then selects a hard disk device on which DSM has been installed.

3. When a user enters <cr> in response to the Operating System Utilities Menu prompt.

4. When a user other than a timesharing user enters a "/" in response to the File Systems Directory prompt.

5. When a user returns to OS-65U from CP/M.

This directory cannot be accessed by a timesharing user, including User 0, because timesharing runs only under OS-65U. There is therefore, no choice to make once timesharing is started.

The effect of taking each selection from the Operating Systems Directory is described below.

0> Operating System Utilities

1. The DSM requests a password. Enter the password of OS-65U File System 0.

2. The Operating System Utilities Menu is displayed.

1> OS-65U

1. The DSM requests a password. Enter the password for the OS-65U Operating System.

2. The OS-65U File Systems Directory is displayed.

2> CP/M

1. The DSM requests a password. Enter the password for the CP/M Operating System.

2. The CP/M Operating System becomes active. See the CP/M Usage and Installation Guide for further information.

NOTE

This selection can be made only from a networked workstation. It is not available to timesharing users.

THE OPERATING SYSTEM UTILITIES MENU

The Operating System Utilities Menu appears as follows:

```
1> Create an operating system
2> Rename an operating system
3> Delete an operating system
4> Print an operating system directory
5> Select a new device

Enter a <Return> to select an operating system.

Your selection?
```

The Operating System Utilities Menu is displayed when Selection #0 is taken from the Operating Systems Directory, and the password for OS-65U File System 0 is entered.

For information on Selection #1, see above under "Create an Operating System". For information on Selection #2, see above under "Modify an Operating System". For information on Selection #3, see above under "Delete an Operating System". All of these headings appear under the major heading "Modifying Operating Systems".

The effect of taking each of the remaining selections from the Operating System Utilities Menu is as follows:

4> Print an Operating Systems Directory

1. The DSM prints:

   Should the directory be printed on a printer (Y/N)?

2. To display the directory on the terminal, reply "N". A directory similar to the Operating Systems Directory will be displayed.

3. To print the directory on a printer reply "Y". The DSM will ask:

   What is the device number of the printer?

   Enter the appropriate device number. A directory similar to the Operating Systems Directory will be printed on the indicated printer. Be sure the printer is connected and ready to print. If it is not, the computer will wait until the printer is available.

4. The DSM prints:

   Do you want to see another directory (Y/N)?

   If you want another directory, reply "Y", then go to Step 1. If not, reply "N".

5. The DSM prints:

   Enter a <Return> to continue?

   Enter <cr>. The Operating System Utilities Menu will be displayed.

5> Select a New Device

This selection can be used only to select a hard disk that has been configured with the Disk Systems Manager. To select any other device, enter Immediate Mode and give a DEV command.

1. The DSM prints on the terminal:

   Device?

   Enter the Device Letter of the desired device.

2. The DSM makes that device the current device, then prints the Operating Systems Directory for the device.

THE FILE SYSTEMS DIRECTORY

A typical File Systems Directory is as follows:

File Systems

No.	Name	Address	Length
0>	OS-65U File Systems Utilities	-	-
1>	Accounting System	1519616	774144
2>	Word Processing System	2293760	114688
3>	Backup WP System	2408448	114688

The File Systems Directory is displayed when #1, the selection for the OS-65U Operating System, is taken from the Operating Systems Directory, and the password for the OS-65U Operating System is entered. It is also displayed when #6, "Select File System", is taken from the BEXEC* Main Menu, and a device is chosen on which the DSM has been installed.

The effect of taking each selection from the File Systems Directory is as follows:

0> File System Utilities

1. The DSM requests a password. Enter the password of OS-65U File System 0.

2. The File System Utilities Menu is displayed.

Any Other Selection

Taking any other selection connects you to the OS-65U file system named in that selection. When such a selection is made:

1. The DSM requests a password. Enter the password of the selected file system.

2. The selected file system becomes the current file system. The standard BEXEC* Main Menu is displayed.

To exit from the File Systems Directory to the Operating Systems Directory, enter a "/" in response to the File Systems Directory prompt.

THE FILE SYSTEM UTIILTIES MENU

The File System Utilities Menu appears as follows:

```
1> Create a file system
2> Rename a file system
3> Delete a file system
4> Print/Display directories
5> Select a new device
6> Transfer programs/files
```

```
 7> OS-65U utilities menu
 8> Transient utilities menu
 9> Multi-User menu
10> Unlock system

Enter a <Return> to select a file system.

Your selection?
```

The File System Utilities Menu is displayed when Selection #0 is taken from the File Systems Directory, and the password for OS-65U File System 0 is entered.

For information on Selection #1, see above under "Create a File System". For information on Selection #2, see above under "Rename a File System". For information on Selection #3, see above under "Delete a File System". All of these headings appear under the major heading "Modifying File Systems".

The effect of taking each of the remaining selections from the File System Utilities Menu is as follows:

4> Print/Display Directories

1. The DSM displays the terminal:

```
1> Print an operating systems directory
2> Print an OS-65U file systems directory
3> Print a directory of an OS-65U system

Select the type of directory you want to see.
```

2. The DSM displays on the terminal:

```
Should the directory be printed on a printer (Y/N)?
```

   If you do not want the directory to go to a printer, reply "N", then go to Step 4.

3. If you do want the directory to go to a printer, enter "Y". The DSM will display on the terminal:

```
What is the device number of the printer?
```

   Enter the appropriate device number.

4. If you selected #3, "Print a directory of an OS-65U system", the DSM displays on the terminal:

```
There are n systems.
You want to see the directory for which 65U system?
```

   where n is the number of OS-65U file systems other than File System 0. Enter the number of the file system whose directory you want to see.

5. The DSM prints the requested directory on the line printer, or displays it on the terminal.

6. The DSM displays on the terminal:

   ```
   Do you want to see another directory (Y/N)?
   ```

   If you want to see another directory, reply "Y" and go to Step 1. If not, reply "N". The File System Utilities Menu will be displayed.

5> Select a New Device

This selection can be used only to select a hard disk that has been configured with the Disk Systems Manager. To select any other device, enter Immediate Mode and give a DEV command.

1. The DSM prints:

   ```
   Device?
   ```

   Enter the Device letter of the desired device.

2. The DSM will make that device the current device, then print the Operating Systems Directory for the device.

6> Transfer Programs/Files

This selection can be used to transfer files from any device or file system to any other device or file system.

1. The DSM prints:

   ```
   Transfer from unit <A>?
   ```

   Enter the letter of the device from which you wish to transfer files.

2. If the device indicated in Step 1 is a hard disk, the DSM prints:

   ```
   1> Transfer by system number
   2> Transfer by disk address and size

   Your selection <1>?
   ```

   Hit <cr>. The DSM will print:

   ```
   There are n systems.

   Transfer from system number <0>?
   ```

   Enter the number of the file system from which you wish to transfer files. The DSM will request the password of the indicated file system. Enter that password.

3. The DSM prints:

```
Transfer to unit <B>?
```

Enter the letter of the device to which you wish to transfer files.

4. If the device indicated in Step 3 is a hard disk, the DSM prints:

```
1> Transfer by system number
2> Transfer by disk address and size

Your selection <1>?
```

Hit <cr>. The DSM will print:

```
There are n systems.

Transfer to system number <0>?
```

Enter the number of the file system to which you wish to transfer files. The DSM will request the password of the indicated file system. Enter that password.

5. The DSM prints:

```
1> Transfer files from S to D.
2> Create files on D if needed.
3> Do both operations.

Your selection <3>?
```

where S is the letter of the source device indicated in Step 1, and D is the letter of the destination device indicated in Step 3.

Selection 1 tells the DSM to transfer only files that already exist on D. Selection 2 tells it to create on D any destination file that does not already exist there but to avoid transferring any data. Selection 3 tells it to create files as needed on D, and to transfer data as well. Enter your choice.

6. If you selected 1 or 3 in Step 5, the DSM prints:

```
Do you want prompting before a file is transferred <No>?
```

If you enter "N" the DSM will transfer files without verifying each transfer. If you enter "Y" the DSM will give you an opportunity to suppress the transfer of each file, by printing a line of the form

```
- Copy "FILENAME" <No>?
```

before each proposed file transfer.

7. If you selected 2 or 3 in Step 5, the DSM prints:

```
Do you want prompting before a file is created <No>?
```

If you enter "N" the DSM will create files as needed on the destination device without verifying each creation. If you enter "Y" the DSM will give you an opportunity to suppress the creation of each file, by printing a line of the form

- Create "FILENAME" <N>?

before each proposed file creation.

8. The DSM prints:

   What is the name of the first file <First File>?

   If you enter <cr>, the DSM will start by transferring the first file in the source directory (after the directory file), then proceed to each subsequent file in turn. If you enter a file name, the DSM will scan the source directory until it encounters a file having that name, transfer the file, then proced to each subsequent file in turn. Enter your response.

9. The DSM prints:

   What is the name of the last file <Last File>?

   If you enter <cr>, the DSM will transfer files until it reaches the last file in the source directory. If you enter a filename, the DSM will transfer files until it has transferred the indicated file, then cease transferring. Enter your response.

10. The DSM prints:

    What is the file name to key on <All>?

    This prompt allows you to transfer from the set of files indicated in Steps 8 and 9 only those files whose names match a pattern you type in.

    To transfer all files regardless of name, enter <cr>. To transfer just the file that has a particular name, enter that name. To transfer only files whose names have a particular pattern, enter a string consisting of ASCII characters and the wild card character "?". A "?" matches ny character including a blank. Some examples:

String	File Transferred
T??	Every file whose name begins with "T" and is 1, 2, or 3 characters long.
W??L	Every file whose name begins with "W", ends with "L", and is 4 characters long.
R?G?R	Every file whose name has an "R" in the first position, a "G" in the third, an "R" in the fifth, and is 5 characters long.

WATER?	Every file whose name begins with the string "WATER" and is 5 or 6 characters long.

11. The DSM prints:

    Are you ready (Y/N) <No>?

    If you reply "N", no files will be transferred. If you reply "Y" file transfer will begin.

12. If, during file transfer, a source file is selected for transfer that is larger than a pre-existing destination file of the same name, the DSM prints:

    ** FROM FILE LARGER THAN TO FILE **

    Do you want to delete "FILENAME" on Device D and re-create it as a larger file <No>?

    Enter "Y" or "N".

13. When all files that meet the criteria established in Steps 8, 9 and 10 have been processed, or if you entered "N" in Step 11, the DSM prints:

    *** End of Execution ***

    Enter <Return> to continue?

    Enter <cr>. The File System Utilities Menu will be displayed.

File transfer can be interrupted at any point by typing Control-C.

7> OS-65U Utilities Menu

Taking this selection is identical in effect to selecting #3, "System Utilities", from the BEXEC* Main Menu. The System Utilities Menu is printed, as follows:

```
1) DIRECtory
2) Print DIRECtory
3) Create file
4) Delete file
5) Rename file
6) Dump file contents
7) Copy from file to file
8) Disk Copier
9) General purpose KEYBASE comaptible file editor
```

See Chapter 2 for information on these selections.

8> Transient Utilities Menu

Taking this selection is identical in effect to selecting #4, "Transient Functions", from the BEXEC* Main Menu. The Transient Utilities Menu is printed, as follows:

1) Editor
2) Resequencer
3) Extended INPUT
4) Common Variables
5) Terminal Setup
6) Standard System

See Chapter 6 for information on these selections.

9> Multi-User Menu

Taking this selection is identical in effect to selecting #7, "Select Multi-User Menu", from the BEXEC* Main Menu. The Multi-User Menu is printed, as follows:

1) Start/Restart Timeshare User(s)
2) Start Network
3) Enable Timeshare User Access to Network
4) Enable Work Station Access to Network
5) Change Network Disk Access Limits
6) Examine/Clear User Semaphores
7) Start Spooling
8) Stop Spooling
9) Examine/Modify Spool Control File
10) Interrupt Despooling
11) Begin Despooling
12) Set Time and Date
13) List Timeshare User(s) Current Active

See Chapters 10, 11, and 12 for information on these sections.

10> Unlock System

Taking this selection is identical in effect to selecting #5, "Enter Immediate Mode", from the BEXEC* Main Menu. OS-65U prints:

```
Password:
```

Enter "UNLOCK". You are not in Immediate Mode, and are connected to File System 0.

THE OS-65U/DSM USER INTERFACE

Use of the Disk Systems Manager causes certain changes in the OS-65U user interface. These changes are as follows:

The Main Menu and the Operating Systems Directory

When the DSM is not used, the BEXEC* Main Menu is at the top of the OS-65U Menu Tree. When the DSM is used, the Operating Systems Directory is the top of the Menu Tree for users operating the system in a single user mode and for networked workstation users.

To get from the Operating Systems Directory to the BEXEC* Main Menu, do the following:

1. From the Operating Systems Directory select #1, the OS-65U Operating System.

2. Enter the OS-65U Operating System Password.

3. The File Systems Directory will be displayed. Select a file system.

4. Enter the password of the file system selected.

5. The BEXEC* Main Menu will be displayed.

The Main Menu and the File Systems Directory

For a timesharing user, the File Systems Directory is at the top of the menu tree when the DSM is used. To get from the File Systems Directory to the BEXEC* Main Menu, do the following:

1. Select a file system from the File Systems Directory.

2. Enter the password for that system.

3. The BEXEC* Main Menu will be displayed.

The Main Menu and the File System Utilities Menu

The File System Utilities Menu is a superset of the BEXEC* Main Menu. It can be thought of as the Main Menu for File System 0. The equivalencies between the File System Utilities Menu and the BEXEC* Main Menu are discussed above under "The File System Utilities Menu".

OS-65U TIMESHARING

This chapter tells:

- What is OS-65U timesharing.
- How to start and restart timesharing.
- How to tell what timesharing users are active.
- How to set and reset the timesharing clock and calendar.
- How to retrieve the date and time.
- How to use the timesharing countdown timer.

This chapter does not tell:

- How to install the hardware required for timesharing. See your OSI dealer for information about hardware installation.
- How to install and configure the software needed for timesharing. That is discussed in Chapter 16.
- How to program in a timesharing environment. Such programming is covered in Chapters 13 and 14.

WHAT IS OS-65U TIMESHARING

OS-65U Timesharing is an extension to single-user OS-65U. With Timesharing, OS-65U becomes a multi-user, real-time operating system which can support the concurrent operation of up to six users. Each user has his own terminal, memory space, operating system, and BASIC interpreter. All users share the computer's CPU, printers, disk drives, and file systems.

OS-65U Timesharing provides:

- An interrupt-driven scheduler clock.
- Interrupt-driven terminal I/O.
- Automatic semaphore-mediated control of the shared CPU and disk drives.
- Commands and semaphores for user management of the shared printers and files.
- A system-wide real-time clock and calendar, and a count-down timer for each user.
- Maximum possible protection of each user from errors made by other users.

OS-65U Timesharing can run on any Masterkey 230 or 250 series computer. A 230 can support up to four timesharing users, called User 0 through User 3. A 250 can support up to six users, called User 0 through User 5. Except for the maximum number of users, there is no difference between timesharing on a 230 and timesharing on a 250.

It is sometimes useful to run OS-65U Timesharing with only one user (User 0). This could happen if the clock, calendar, or count-down timer were the only function wanted from the timesharing mode. This section assumes that there are several timesharing users, as will usually be the case in practice.

Timesharing Partitions

A Masterkey timesharing computer is divided into partitions.

A partition consists of:

- 48K of user memory
- A copy of OS-65U
- A copy of the OS-65U BASIC interpreter
- A terminal

There is a separate partition for each timesharing user. Each partition is numbered, and its number is the inverse of that of the user that runs on it. That is, User 0 runs on Partition F, User 1 on Partition E, etc.

A timesharing user is started (booted and given access to the resources of the computer) by starting his partition, as described in the next section.

User 0

Each Masterkey timesharing computer has a privileged user: the one who is running on Partition 0. This user is known as User 0.

When a timesharing computer is first booted, only User 0 is started. User 0 can then start any or all other timesharing partitions. Only User 0 can start a timesharing partition.

STARTING THE TIMESHARING SYSTEM

The following are the steps needed to start timesharing. They assume that all required timesharing hardware and software has been installed and configured.

Start the Computer

If the computer is not already running, start it and boot it from the hard disk as described in Chapter 2. The computer will come up in single-user mode on File System 0 and print the OS-65U Main Menu.

NOTE

Do not attempt to boot the system directly from the Timesharing Diskette. The bootable system portion of the diskette contains MPUTIL, a routine for checking the integrity of the computer's timesharing partitions, rather than OS-65U. MPUTIL is for use when installing or troubleshooting an OS-65U Timesharing system.

Activate the Timesharing System

From the Main Menu, select #7, "Select Multi-User Menu". The Multi-User Menu looks as follows:

```
 1. Start/Restart Timeshare User(s)
 2. Start Network
 3. Enable Timeshare User Access to Network
 4. Enable Work Station Access to Network
 5. Change Network Disk Access Limits
 6. Examine/Clear User Semaphores
 7. Start Spooling
 8. Stop Spooling
 9. Examine/Modify Spool Control File
10. Interrupt Despooling
11. Begin Despooling
12. Set Time and Date
13. List Timeshare User(s) Currently Active
```

From the Multi-User Menu select #1, "Start/Restart Timeshare User(s)". The timesharing system will become active and request the information it needs to start timesharing.

Bring Up Timesharing Partitions

The system prints:

Bring up all users (A) or a specific one(S)?

Enter an "A" to immediately start all user partitions. The system will print a confirmation for each partition that it starts, or an error message for any partition that it cannot start.

Once all partitions have been started, or if you entered an "S", the system prints:

Bring Up a Specific User (1-15) or Done (0)?

If you responded "A" to the previous prompt, you are done and should enter a "0". Otherwise, enter the number of the first user partition you want to start. The system will start that partition (if possible) and print a message, as with the "A" response above. It will then repeat the specific-user prompt. This cycle continues until you enter a "0".

If you try to bring up a partition that does not exist, the system takes no action, and repeats the specific-user prompt.

Set the System Date and Time

The system next prints on the console:

Set date and time?

If you want to set the date and time, enter "Y". Otherwise, enter "N" or <cr>.

If you enter "Y" the system prints:

Date (M,D,Y)?

Enter the current date as the numeric month, the numeric day, and the last two digits of the year. For example, if the date is July 4, 1984 enter 7,4,84.

When you have entered the date, the system prints:

Time (H,M,S)?

Enter the time as the numeric hour, minute, and second. Give the hour in 24-hour form. For example, if the time is 1:00:04 P.M., enter 13,00,04.

After you have entered the time, or if you replied to the "Set date and time?" prompt with "N" or <cr>, the system prints:

Enter a <return> to continue?

Type a <cr>; the Multi-User Menu will be displayed.

The system is now up and running in timesharing mode. Programming can begin on all partitions that have been started.

TIMESHARING ENVIRONMENT AFTER STARTUP

As each timesharing user's partition is started, the main menu is displayed on his terminal. The menu header indicates that the user is in timesharing mode, and gives his partition (user) number.

When a partition is started, it is given its own copy of OS-65U. This copy is not read from the disk, but is copied directly from the memory space of User 0. Therefore:

1. Any transient utility enabled for User 0 when a timesharing partition is started will be enabled for that partition as well.

2. When a timesharing partition is started, it will expect the same type of terminal that User 0 has. If some other type of terminal is connected to the partition, the Terminal Setup utility must be used to give OS-65U the correct terminal type. Terminal Setup is discussed in Chapter 6.

RESTARTING A TIMESHARING USER

Occasionally a timesharing user will make or experience an error that locks up his partition completely. That user can be restarted as follows:

1. On the console terminal (User 0) run the timesharing startup program as described above under "Activate the Timesharing System".

2. Start only the user partition that has locked up.

3. Reply <cr> to the date-and-time prompt.

The locked-up user will be restarted and receive the main menu. His workspace will be cleared by the restart process, just as if he had been a single user and had rebooted his system. Other users will not be affected, and the system date and time will remain unchanged.

Only User 0 can start or restart timesharing. Therefore, if user 0 locks up, the system must be rebooted in order to restart him. All other users should terminate any running programs that modify files, and SAVE any programs that have been modified since they were LOADed. The system can then be rebooted, and timesharing restarted.

Very rarely, an error will occur that locks up the entire system. If this happens, redo the steps given above under "Starting the Timesharing System". All user workspaces will be cleared.

Repeated lockups of one or all users may indicate a hardware problem. If you experience repeated lockups for which there is no obvious software explanation, contact your OSI dealer.

LISTING ACTIVE TIMESHARING USERS

To find out what users are currently started in timesharing mode, access the Multi-User Menu and select #13, "List Timeshare User(s) Currently Active". The system will print a list of all currently active users, then print

Start/restart any users?

If you want to start or restart any users, answer "Y", then proceed as described above under "Restarting a Timesharing User". If you don't want to start or restart any users, answer "N" or <cr>.

USING THE CLOCK AND CALENDAR

The clock and calendar store the time and date in numeric form. They are set when timesharing is started, as described above. Once set, they are maintained by the system until power is turned off or the reset switch is pressed. Restarting any or all timesharing users does not affect the clock and calendar unless the time and date are explicitly reset.

Setting and Resetting the Time and Date

The time and date can be set or reset whenever desired. Access the Multi-User Menu and select #12, "Set Time and Date". The system will ask the same questions that it does when the date and time are set as part of timesharing and startup, and expects the same responses.

Retrieving the Time and Date

The clock keeps the time in the form seconds, minutes, hours. Seconds are maintained with 1 second resolution. Hours are kept in 24-hour form. Second, minute, and hour are kept in locations 55919, 55920 and 55921 respectively. The following BASIC code could be used to retrieve the time.

```
10  A  =  55919
20  S  =  PEEK(A) : REM SECONDS
30  M  =  PEEK(A+1 ) :REM MINUTES
40  H  =  PEEK(1+2) : REM HOURS
```

The calendar keeps the date in the form day, month, year. The months are numbered 1-12 for January-December, and the year is stored as the last two digits of the numeric year. Day, month and year are kept in locations 55922, 55923 and 55924 respectively, and are read by PEEKing those locations. The following BASIC code could be used to retrieve the date for use in a program:

```
10  A  =  55922
20  D  =  PEEK(A) : REM DAY
30  M  =  PEEK(A+1) : REM MONTH
40  Y  =  PEEK(A=2) : REM YEAR
```

USING THE COUNTDOWN TIMER

The countdown timer accepts a time duration given in hours, minutes and seconds. When the specified duration has elapsed, the timer runs the program RTMON. There is a separate countdown timer for each timesharing user.

The countdown timer can be set for any duration from 1 second to 255 hours, 59 minutes and 59 seconds. To set the timer, three POKES are required, as follows:

```
POKE    8906,    number-of-seconds
POKE    8907,    number-of-minutes
POKE    8908,    number-of-hours
```

To start the timer running, execute

```
POKE 8909, 1
```

To stop the timer, execute

```
POKE 8909, 0
```

The timer can be started and stopped in this way as often as needed.

When the countdown timer reaches 0, it executes POKE 8909, 0, interrupts any program being executed by the user who started the timer, and loads and runs the system program RTMON. As supplied by OSI, RTMON just prints a message stating that it has run, and leaves the user in Immediate Mode.

The countdown timer does not know or care what RTMON does. It just finds a program by that name and runs it. RTMON may therefore be modified as needed, or replaced entirely with some other RTMON.

There is only one copy of RTMON for all timesharing users. If all users want the same results when RTMON executes, the program can be rewritten to produce that result. If different users want different results from RTMON, the program must determine what user is executing it, then jump to code specific to that user. RTMON can determine who is running it by PEEKing location 55381. For example, a multi-purpose RTMON could be organized as follows:

```
100     A = PEEK (55381)
200     IF  A = 0  GO TO 1000
300     IF  A = 1  GO TO 2000
400     IF  A =2  GO TO 3000
500     IF  A = 3  GO TO 4000
1000    REM RTMON CODE FOR USER 0
  .
  .
  .
  .
1500    GO TO 10000
2000    REM RTMON CODE FOR USER 1
  .
  .
  .
  .
  .
  .
2500    GO TO 1000
3000    REM RTMON CODE FOR USER 2
  .
  .
```

```
           .
           .
           .
           .
3500       GO TO 10000
4000       REM RTMON CODE FOR USER 3
           .
           .
           .
           .
           .
           .
4500       GO TO 10000
10000      END
```

OS-65U PRINT SPOOLING

This chapter discusses:

- What is OS-65U Print Spooling
- How Print Spooling Works
- How to start the spooler
- How to stop the spooler
- How to start the despooler
- How to control and stop the despooler
- How to control the Spool Job Manager

This chapter does not tell you how to install and configure the software needed for Print Spooling. That is covered in Chapter17.

WHAT IS OS-65U PRINT SPOOLING

OS-65U Print Spooling is a method for redirecting printer-directed program output to a disk file, then printing that file when it is complete. The process of writing the output to the file is termed spooling. The process of printing the file is termed despooling. Be careful not to confuse spooling, the writing of data to a spool file, with Print Spooling, which encompasses both spooling and despooling.

Without Print Spooling, a program could not write to a printer unless the printer were currently free, an unacceptable restriction in a multi-user environment. With Print Spooling, a program can perform printer-directed output independently of printer availability.

No special programming is required to use Print Spooling. Whenever an OS-65U program needs to write to the line printer, it writes to Device #5. If print spooling is not occurring, the output will be printed immediately on Device #5. If print spooling is being done, the output will be written to a spool file instead. The program is exactly the same in either case. The printer commands FLAG 100 and FLAG 101 may be executed under print spooling, and Device #5 paging operates as usual.

Ordinarily, Print Spooling is controlled by taking menu selections. When the Common Variables transient utility is enabled, spooling can be started and stopped as desired by an executing program. Despooling, however, cannot be started or stopped from within a program.

Both spooling and despooling are done from Device E. A timesharing user spools and despools from Device E of the computer that contains his partition. A workstation user spools and despools from Device E of his control station. A stand-alone user, or a user on a workstation not currently linked to the network, spools and despools from his own, local Device E.

When a Device E contains multiple file systems, Print Spooling is implemented and performed separately for each system. Every file system on which Print Spooling will be done must contain all the files that constitute the Print Spooler (these are listed below). Spooling and despooling are always done from the file system most recently selected (the current file system) at the time spooling or despooling was started.

Print Spooling can run on a 230C series computer only when the computer is acting as a workstation, since only then is a Device E available to it.

HOW PRINT SPOOLING WORKS

Print spooling is controlled by the Spool Job Manager, which is part of OS-65U. The Spool Job Manager has access for a set of pre-existing files called spool files, which hold data that has been spooled but not yet printed. There may be from 3 to 15 such files, named SPL1, SPL2,...,SPL15, on a given file system. These files are created when Print Spooling is installed and configured. The Manager keeps track of these spool files by accessing and storing information in another file, the Spool Control File, whose filename is SPL 0.

Spooling

Spooling is started and stopped on a per-user basis. When spooling is started by a user, the Spool Job Manager finds an empty spool file and opens it. The Manager thereafter writes to the open spool file any data directed by the user to Device #5. When spooling is stopped by a user, the Manager closes the spool file and marks it FULL, i.e. ready to print.

Spooling is performed by a program called the spooler. The spooler overlays the OS-65U floppy disk driver, with the result that a computer or timesharing partition that is spooling cannot access its floppy disk(s). The floppy disk driver is automatically reinstalled when spooling ends. The spooling program also overlays the space used by the #6 (WPDRIV) Word Processing Printer Driver, with the result that the Word Processing Printer cannot be installed while spooling is active. The option to install this printer driver is restored when spooling ends.

Despooling

Despooling is started and stopped for all users at once. When despooling first starts, the Spool Job Manager checks each spool file (via data in SPL 0) and prints on Device #5 any file marked FULL. When the file has been printed, it is marked EMPTY, i.e. ready for re-use. The Manager continues to check spool files in round-robin order as long as it is active. When a spool file becomes FULL after despooling is already underway, it will be printed as soon as its turn comes.

Despooling is done by a program called the despooler. When despooling begins, the computer or timesharing partition that started it becomes dedicated to running the despooler program. The dedicated computer or partition is then unavailable for programming, and will remain unavailable until despooling ends. Its terminal becomes the Despooler Console, and displays the name of each file that is printed.

When the despooler is stopped, the despooler ceases to print spool files, and spooled data accumulates until despooling is restarted.

Despooled output always appears on the printer of the system doing the despooling. Thus if a workstation begins despooling, the workstation's printer will be used, while if a timesharing user begins despooling, the printer of the computer that contains his partition will be used.

AUTOMATIC DESPOOLER STARTUP

A control station can be configured to start despooling automatically on a preselected timesharing partition as soon as timesharing begins. Automatic despooler startup can be enabled on one control station and disabled on the other, and can use different timesharing partitions on the two control stations. Automatic despooler startup can run only on a timesharing partition other than Partition 0.

When automatic startup is used, a terminal need not be connected to the partition on which the despooler is run. If one is, it will be the Despooler Console.

For information on how to enable Automatic Despooler Startup, see Chapter 17.

STARTING THE SPOOLER

The spooler can be started via the menu system, or by an executing program. The spooler can be started by a program only if the Common Variables transient function is enabled.

To start the spooler by using menus, access the Multi-User Menu and select #7, "Start Spooling". The system will reply:

```
Spooling Enabled
```

Henceforth, any data written to Device #5 will be stored in a spool file rather than being printed on the line printer.

To start the spooler from within a program, first set the following common variables:

PR$	=	name of program to transfer to after the spooler has been started.
PW$	=	password of program to transfer to

LN	=	line in program to transfer to
ST$	=	spool job title

Then execute:

```
RUN ["SPOOL1", "PASS", 10]
```

Spooling will begin just as if the menu system had been used. Once spooling is underway, control will pass to program PR$ at line LN. Typically, PR$ is the same program that ran SPOOL1, and LN is the line after the RUN statement, so that the RUN statement is equivalent to a procedure call. Another possibility is to make the RUN statement activate a third program, then to spool data from that program.

Use of ST$ is optional when SPOOL1 is run from within a program. ST$ can be at most 32 characters long. If no ST$ has been defined, PR$ will become the spool job's name.

Once you have started spooling in a file system, you cannot leave that file system until you stop spooling.

STOPPING THE SPOOLER

The spooler can be stopped via the menu system, or by an executing program. The spooler can be stopped by a program only if the Common Variables transient function is enabled.

To stop the spooler by using menus, access the Multi-User Menu and select #8, "Stop Spooling". The system will reply:

```
Print Spooling Disabled
```

Henceforth any data written to Device #5 will be printed immediately.

To stop the spooler from within a program, set PR$, PW$ and LN as when starting spooling (above), then execute:

```
RUN ["SPOOL2", "PASS", 10]
```

Spooling will stop just as if the menu system had been used, and control will pass to program PR$ at line LN.

STARTING THE DESPOOLER

The despooler cannot be started by an executing program. It can be started via menu on any workstation, timesharing partition, or stand-alone computer; or it can be configured to start automatically on a timesharing partition whenever timesharing begins.

To start the despooler by using menus, access the Multi-User Menu from the terminal of the computer or partition that is to run the despooler. Timesharing Partition 0 should not be used, because it would then be unavailable for restarting timesharing users. From the menu, select #11, "Begin Despooling". The despooler will then become active. The system it was started on will be unavailable for programming until despooling ends. The despooler will print on its terminal the name of each file that is despooled, or the message "Waiting for Next File" when there are no files to despool.

If Automatic Despooler Startup is used, no action is necessary to start the despooler. It will start automatically as soon as timesharing begins.

When a Device E has more than one file system, the despooler prints files only from the file system that was current when despooling began. When the despooler is started via menu, select the appropriate file system before starting the despooler. When Automatic Despooler Startup is used, select the appropriate file system before starting timesharing.

WARNING

Despooling should not be started on more than one partition of a given timesharing computer, or on more than one workstation connected to a given control station, or on both a workstation and a timesharing partition of a control station. If this rule is broken, loss of spooled data may occur.

CONTROLLING THE DESPOOLER DIRECTLY

There are two ways to control or stop the despooler:

1. Directly, from a terminal connected to the system on which the despooler is running.

2. Indirectly, by accessing the Spool Job Manager's Operator Interface from a system other than the one on which the despooler is running.

This section covers direct control. Indirect control is covered in the next section under "Controlling the Despooler Indirectly".

Direct Control

To control or stop the despooler from a terminal connected to the system on which it is running, type control-C on that terminal. The system's response depends on whether a file is or is not being printed when the control-C is typed in.

If no file is being printed when control-C is received, the despooler will stop immediately. The system will print the message:

Despooling Operation Terminated.

The system on which the despooler was running is now available for programming.

If a file is being printed when control-C is received, the system will suspend printing, then display a Despooler Control Menu. That menu is:

1 - Abort the print job.
2 - Restart the print job from the beginning.
3 - Continue with print.
4 - Terminate despooling operation.

The result when each of these Despooler Control options is selected is as follows:

1 - Abort the print job: Printing is aborted, the printer does an EJECT, and the despooler continues searching for files to print. The status of the spool file that was being printed is set to "A". See "Controlling the Spool Job Manager" below for the significance of this status and information on what to do about it.

2 - Restart the print job: The printer does an EJECT, and the file is reprinted from the beginning.

3 - Continue with print: Printing resumes. The printout is in no way affected by the fact that printing was temporarily suspended.

4 - Terminate despooling: Printing is aborted, the printer does an EJECT, and the despooler is stopped. The partition that was dedicated to despooling becomes available for programming. If a spool file was printing when control-C was entered, its status is left at "B". See "Controlling the Spool Job Manager" below for the significance of this status and information on what to do about it.

After 1, 2, or 3 is selected, the despooler continues to operate. Control-C may be entered again if additional despooler control is desired. After 4 is selected, the Multi-User Menu is displayed.

CONTROLLING THE SPOOL JOB MANAGER

The Spool Job Manager, which manages both spooling and despooling, has an operator interface which allows a user to control various aspects of the Print Spooling process. This interface allows you to:

1. Alter the status of any spool file.

2. Examine and change any parameter of any spool file.

3. Abort the current print job.

4. Restart the current print job from the beginning.

5. Suspend printing of the file currently being despooled (if any).

6. Stop the despooler.

The Spool Job Manager interface cannot be accessed from a system that is currently running the despooler, because that system is entirely dedicated to running the despooling program. The interface can be accessed from any other workstation or timesharing partition.

To use the Spool Job Manager operator interface, first enable Extended Input as described in Chapter 6. The operator interface is available only when Extended Input is enabled.

Next, access the Multi-User Menu and select #9, "Examine/Modify Spool Control File". The system will then display the Spool Job Manager Display, which shows the current job name and status for each spool file, and contains a menu for use in controlling the Spool Job Manager.

Format of the Spool Job Manager Display

The Spool Job Manager Display begins with 15 lines, one for each of the files SPL1-SPL15 which are used to hold spooled data. In each line, there are three fields:

- Number: 1-15, designating the spool files SPL1-SPL15.
- Job Name: The job name, if any, assigned to the spool file when spooling began. If none was assigned, the Job Name field is blank.
- Status: The current status of the spool file. The possibilities are:

  "E" - Empty. The file is available to receive spooled data.

  "L" - Loading. The file is now receiving spooled data.

  "F" - Full. The file contains spooled data, spooling has been stopped, but the despooler has not begun printing the file.

  "B" - Busy. The file contains spooled data that has been printed in part. When the file has been completely printed, its status will be changed to "E". If the file is never completely printed, perhaps because the despooler was stopped before printing was complete, the file's status will remain "B" until it is reset as described in the next section.

  "A" - The file contains data and was being printed, but printing was stopped because "Abort the Print Job" was selected from a Despooler Control Menu. The status will remain "A" until it is reset as described in the next section.

  "Blank" - The file was not created when Print Spooling was installed.

After the 15 status lines, the Spool Job Manager Display presents a menu, called the Spool Job Manager Menu. That menu is:

1 - Change Status
2 - Printer Control
3 - Examine Job
4 - Exit?

These selections are explained in the following sections.

Changing Spool File Status

To change the status of a spool file, first select #1, "Change Status", from the Spool Job Manager Menu. The system will respond:

Enter number of the job to change:

Enter the appropriate number from Column 1 of the Status display. The system next prints:

Enter new status:
(E)mpty, (B)usy, (L)oading, (F)ull:

Enter the desired status, "E", "B", "L", or "F". The file's status will be changed and the Spool Job Manager Display will be repeated.

The usual reason for changing a spool file's status is that it is "A" because printing of the file was aborted, or "B" because the file was printing when the despooler was stopped. If you want to make the file available to the spooler for re-use, change it's status to "E". If you want to print the file again from the beginning, change its status to "F"; it will then be reprinted next time the despooler scans its entry in the Spool Control File. Another reason to change a spool file's status is that multiple copies are needed. Every time a printed file's status is changed from "E" to "F", it will be printed again.

CAUTION

Do not change any spool file status that is given as "Blank". The status "Blank" means that the file was not created when print spooling was installed.

Examining and Changing Spool File Parameters

The Spool Job Manager Display does not show all the parameters of each spool file, because there is not enough room for them on the screen. To examine all parameters of a spool file, first select #3, "Examine Job", from the Spool Job Manager Menu. The system will respond:

Enter password:

Type in "PASS". The system will respond:

```
Enter number of job to examine:
```

Enter the appropriate spool file number, 1-15.

The system will then print a Spool File Parameter Display. A typical display is:

```
1  FILE NAME             SPL1
2  STATUS                B
3  PRIORITY              9
4  JOB NAME              PRINT.FIL
5  OUTPUT DEVICE         5
```

The meanings of these fields are:

- FILE NAME - The name of the spool file, SPL1-SPL15.
- STATUS - Same as in the Spool Job Manager Display.
- PRIORITY - Unimplemented as of Version 1.44 Always set to 9.
- JOB NAME - Same as in the Spool Job Manager Display.
- OUTPUT DEVICE - Unimplemented as of Version 1.44. Always set to 5.

After displaying the spool file parameters, the system prints:

```
Enter number of field to change:
```

If you do not want to change any parameters, enter a <cr>. The Spool Job Manager Display will be repeated.

If you want to change a field, enter the number shown for it on the parameter list. The system will reply:

```
Enter new field contents: XXXX
```

where XXXX is the current value of the field. This value can be replaced by first deleting it (DELETE key) and then typing in the new value, or it can be modified using the Extended Input capabilities discussed in Chapter 6. The value shown on the screen when <cr> is typed becomes the new value of the field.

After a change has been entered, the Spool File Parameter Display will be repeated, followed again by "Enter number of field to change:". You can now enter another change, or type <cr> to return to the Spool Job Manager Display.

CAUTION

Does not change the file name of a spool file. If the name is changed, the file will be unavailable for spooling or despooling until the correct name is restored.

Do not change any field whose value is "Blank", or assign the value "Blank" to any field. The value "Blank" is for use only when Print Spooling is installed or configured.

Controlling the Despooler Indirectly

The following is the method for indirectly controlling or stopping the despooler. The method for controlling it directly is given above under "Directly Controlling the Despooler".

To indirectly control or stop the despooler, first select #2, "Printer Control", from the Spool Job Manager Menu. The Spool Job Manager will display a Despooler Control Menu, as follows:

```
1     -  Abort the print job.
2     -  Restart the print job from the beginning.
3     -  Pause printer.
4     -  Terminate Despooling Operation.
<cr>  -  Return to Spool Job Manager.
```

The result when each of these Despooler Control options is selected is as follows:

1 - Abort the print job: Printing stops, the printer does an EJECT, and the despooler continues searching for files to print. The status of the spool file that was being printed is set to "A".

2 - Restart the print job: The printer does an EJECT, and the file is reprinted from the beginning.

3 - Pause Printer: Printing stops, and remains stopped until a <cr> is typed in.

4 - Terminate Despooling: Printing stops, the printer does an EJECT, and the despooler is stopped. The partition that was dedicated to despooling becomes available from programming. If a file was printing when this selection is made, its status is left at "B".

After 1, 2, or 3 is selected, the despooler continues to operate. After any selection, the Spool Job Manager prints:

```
Enter <Return> to continue with spool manager?
```

Enter a <cr>. The Spool Job Manager Display will be repeated. This <cr> is also the <cr> that causes printing to resume after "Pause Printer" has been selected.

PRINT SPOOLING ERRORS

The following errors can occur during Print Spooling. If the error occurs when a selection related to Print Spooling has been taken from the Multi-User Menu, the text shown appears on the terminal. If the error occurs when an executing program attempts to start or stop spooling, the text shown appears in the common variable ER$.

"Must disable WPDRIV to enable spooling." An attempt to start spooling was made while WPDRIV was enabled. WPDRIV and the spooler cannot run at the same time. Instructions for disabling WPDRIV are in Chapter 8.

"No spool file available.": An attempt to start spooling has been made, but no spool file was available to hold the data. Possible solutions are:

1. If there are full spool files, despool them, or if the spooler is already running, wait for them to be despooled.

2. If there are spool files that are stuck in an intermediate state, reclaim them. This is discussed above under "Controlling the Spool Job Manager".

3. If not enough spool files were created when Print Spooling was installed and configured, create more as described in Chapter 17, "Installation of OS-65U Print Spooling Software".

"Spooling already enabled.": An attempt to start spooling has been made, but spooling had already been started.

"Spooling not enabled.": An attempt to stop spooling has been made, but spooling had not previously been started.

"Spooling not installed.": Print spooling has not been installed on the current file system. See Chapter 17, "Installation of OS-65U Print Spooling Software", for instructions on how to install Print Spooling.

System Error During Spooling

If the spool file into which spooled data is being placed becomes full, Error 132 occurs, the current program (if any) is aborted, and the user is left in Immediate Mode. No message is placed in ER$, and the spooler continues to run.

The data in the spool file is not lost when this error occurs. The file can be despooled by setting its status to "F", as described above under "Controlling the Spool Job Manager", and starting the despooler if it is not started already.

If this error occurs often, the existing spool files are too small for the application(s) that use them. The solution is to replace them with larger files, as described in Chapter 17 under "Modifying the Print Spooling Utility".

OS-65U NETWORKING

This chapter contains the information you need in order to start and use a OS-65U Network.

This chapter discusses:

- What a OS-65U Network is.
- Controller nodes
- Workstations
- Data access in a network
- How to start and restart networking
- Accessing a started network
- Monitoring network activity

This chapter does not discuss:

- How to install the hardware required for networking. See your OSI dealer for information about hardware installation.
- How to install and configure the software needed for Networking. That information is in Chapter 18.
- How to program in a network environment. That is discussed in Chapters 13 and 14.

WHAT IS OS-65U NETWORKING

OS-65U Networking is an extension to OS-65U Timesharing. With networking, OS-65U becomes a distributed system which can link up to 18 OSI computers into a computer network. The network can have one or two primary nodes, called controllernodes, each of which can serve up to eight secondary nodes, called workstations. In addition, each controller node can run a full complement of timesharing users.

This chapter assumes a network with two controller nodes, each of which has both workstations and timesharing users. However, the chapter applies to a network of any size. A smaller network just has fewer nodes and users -- it does not behave any differently.

OSI computers are connected into a network by wiring them together physically, then giving network startup commands (described below) to each one. Once a network is functioning, any workstation can access data and programs on Device E (the hard disk) of either controller node, as well as the data and programs on its own floppies. The controller nodes can also access each-other's Device E's. The purpose of networking is to permit this sharing of data and programs.

The fact that the computers in a Keyring network are physically wired together does not impair their ability to function separately. A computer wired up as a controller node can be used in single-user mode by booting it but not starting timesharing or networking; as a timesharing computer by booting it and starting timesharing, or as a controller node by booting it and starting both timesharing and networking. A timesharing user on a controller node need not connect to the network; such a user experiences the same environment that he would if his control station were doing only timesharing. A workstation user also need not connect to the network. If he does not, his computer will behave just as it would if no network existed.

CONTROLLER NODES

A controller node can be any hard-disk-based Masterkey 230 or 250 computer. A controller node has three functions:

1. It handles all network traffic passing through it, routing each request and response to the proper destination.

2. It acts as a central data store. Any node in the network can access programs and data on a controller node's Device E.

3. It performs all processing for any timesharing users directly connected to it.

A controller node is most frequently a computer with a single hard disk (Device E). A computer with two hard disks (Device E and Device F) can be used, but only Device E will be accessible to network users.

The Network Manager

An operating controller node must be in timesharing mode with at least one started timesharing partition. This partition runs the Network Manager, the program that controls and co-ordinates network processing. The partition running the Network Manager is unavailable for user programming while networking is in progress. Its terminal becomes the Network Console, on which network messages are displayed.

There are two ways to select the timesharing partition that will run the Network Manager:

1. By using its terminal to give the commands that start up the network. This method is discussed below under "Starting the Network".

2. By using Automatic Network Startup.

Automatic Network Startup

A controller node can be configured to start networking automatically on a preselected timesharing partition as soon as timesharing begins. Automatic Network Startup can be enabled on one controller node and disabled on the other, and can use different timesharing partitions on the two controller nodes. Automatic startup cannot use timesharing partition 0.

When automatic startup is used, a terminal need not be connected to the partition on which the Network Manager is run. If one is, it will be the Network Console.

For information on how to enable Automatic Network Startup, see Chapter 18.

WORKSTATIONS

A workstation can be any floppy-disk-based or hard-disk-based 230 or 250 Masterkey Computer. A workstation is most frequently floppy-disk-based. A hard-disk-based computer can be used as a workstation, but its hard disk(s) will not be accessible in any way while the computer is connected to the network.

A workstation has the same capabilities that it would have in stand-alone mode. Networking affects only the data a computer can access, not the tasks it can perform.

A workstation cannot handle timesharing users. A given computer may be used as a workstation, or remain independent and serve timesharing users, but not both.

DATA ACCESS IN A NETWORK

A user who has connected to a network can access:

1. Device E of either controller node.

2. The floppy disks of the computer to which his terminal is wired.

A user connected to a network cannot access

1. Any workstation's Device E

2. The Device F of any computer other than the one to which his terminal is wired.

3. The floppy disks of any computer other than the one to which his terminal is wired.

A user who could connect to a network but has not done so can access the same data that he could if there were no network.

Accessing Remote Data

Accessing of remote data in a network is done in the same way that local data is accessed in a stand-alone Masterkey computer: by naming the device holding the data in a DEV command.

When networking is installed, a Network Device Letter is assigned to Device E of each control station. When the network is running.

1. Any node in the network can access either control station's Device E by giving its Network Device Letter in a DEV command.

2. A workstation can also access its own control station's Device E by specifying "E" in a DEV command.

3. A controller node can access its own Device E just as it could if there were no network.

STARTING THE NETWORK

The following instructions assume that all required hardware and software has been installed and configured.

Starting a Keyring network involves

1. Starting its controller nodes

2. Access of the network by timesharing users

3. Starting its workstations

4. Access of the network by workstations users

Starting a Controller Node

The following steps should be performed to start a controller node when Automatic Network Startup is not in use.

1. Perform all steps shown under "Starting the Timesharing System" in the previous chapter. Both controller nodes must be running in timesharing mode before networking can begin. Only one timesharing partition need actually be started-the one that will run the Network Manager. (See "Controller Nodes", above, for more on the Network Manager.)

2. On the terminal of the timesharing partition that will run the Network Manager, access the Multi-User Menu and select #7, "Start Network".

3. The computer will print a verification request:

    Starting Network will dedicate this user to Networking.

    Are you sure you want to start networking?

    If for any reason you don't want to start networking after all, enter "N". To start networking, enter "Y".

When the verification request is answered "Y", the system prints a message stating that networking has begun, followed by a message giving the controller node's Network Device Letter. The partition used to start the network is now running the Network Manager, and is unavailable for programming. Its terminal has become the Network Console, on which network messages are displayed.

If a controller node is to serve both workstations and timesharing users, the network should not be started on Partition 0. If Partition 0 were dedicated to the Network Manager, it would be unavailable for restarting timesharing users, an operation that can be performed only from Partition 0.

Starting a Controller Node Automatically

When Automatic Network Startup is used, a controller node is started by performing Step 1 above. The rest will be taken care of by the system. The partition used by the Network Manager will become unavailable for programming, and its terminal (if any) will become the Network Console, as with manual startup.

Timesharing Access of the Network

The following steps should be performed by each timesharing user who wants to access the network. The user is assumed to have been started in Step 1 of "Starting a Controller Node", above.

1. Access the Multi-User Menu from the user's terminal.

2. Select #3, "Enable Timeshare User Access to Network"

The system will respond

```
READY
```

The timesharing user can now access the network.

Starting a Workstation

Starting a computer that is a workstation is exactly the same as starting the same computer for stand-alone use. Consult the Masterkey 230 or 250 User Guide and the OS-65U Reference Manual for details.

Workstation Access of the Network

The following steps should be performed by each workstation user who wants to access the network. The workstation is presumed to have been started in single-user mode. It must have a copy of L2NET, the Network Access Program, on the current Device and File System.

1. Access the Multi-User Menu.

2. Select #4, "Enable Workstation Access to Network".

The workstation will respond:

```
PLEASE WAIT
WILL NOW ACCESS NETWORK SYSTEM
```

The network can now be accessed by the workstation.

RESTARTING NETWORKING

If a workstation must be restarted, repeat the sequence given above under "Starting a Workstation". The user's workspace will be overwritten by the restart.

If a timesharing user must be restarted, repeat the sequence given above under "Starting a Network Timesharing User". The user's workspace will be overwritten by the restart.

If the whole network or timesharing system must be restarted, perform the complete sequence given above under "Starting the Network". All users' workspaces will be overwritten by the restart.

MONITORING NETWORK ACTIVITY

When a network is operating, each Network Manager can print a variety of messages describing network activities. These messages appear on the Network Console. The available message types are:

- Transaction messages - printed whenever the Network Manager performs a task for the user.
- Error messages - printed whenever an error occurs in a network.
- Semaphore error message - printed whenever a program tries to set a semaphore that was already set, or clear a semaphore that was already clear. These are "expected error" messages, in that they occur normally during programming in the OS-65U Multi-User Environment.

Semaphore and transaction messages are useful mostly for helping to diagnose network problems. They can be enabled or disabled when the network software is configured (see Chapter 18). They are usually disabled, because when enabled they generate substantial network overhead and thereby impair network throughput.

Network Message Format

The format for all three message types is essentially the same. The format and two examples of these messages are shown below,followed by an explanation of each field.

User Type	User Number	Source Node	Msg. Type	Dest. Node	Disk Addr. Sem. No.	Time
TS	3	K	RQ	M	25088	5:17
L2	0	L	WFR	N	205	6:10

User Type: TS - Timesharing user
L2 - Workstation User

User Number: 0-5, for User 0 - User 5.

Source Node: Node to which the user is connected.

Message Type:

RQ	-	Request to read a data block (3584 bytes)
RD	-	Read a block from disk
RC	-	Receipt of a block that was read
SE	-	Sending a block to be written
WR	-	Writing a block to disk
CF	-	Confirming a block written to disk
WFR	-	WAIT FOR request
WFC	-	WAIT FOR confirm
WCR	-	WAIT CLEAR request
WCC	-	WAIT CLEAR confirm

Destination node: Node from/to which the user is transferring a data block, or on which the semaphore is located.

Disk Address: Disk address from/to which a data block is to be transferred.

Semaphore Number: For WAIT messages, the number of the semaphore.

Time: System time when the message was processed, in minutes:seconds.

Error Messages: An error message is preceded by the following:

***ERROR number port

where number is the error number (see Appendix D) and port is the number of the port to/from which the transfer was occurring when the error was detected.

A semaphore error message will have an error number of 239, and a message type of WFR or WCR.

INTRODUCTION TO MULTI-USER PROGRAMMING

This chapter is intended for those who have written single-user programs, but who do not feel completely familiar with the special programming problems that arise in a multi-user environment.

This chapter discusses:

- What is Multi-User Programming
- Concurrency and Currency Problems
- Preventing Concurrency Problems
- Resource Locks
- Semaphores
- Locking Resources with Semaphores
- Deadlock
- Preventing Deadlock

This chapter is primarily conceptual in orientation. It discusses things true of any multi-user environment. Specific methods for multi-user programming under OS-65U are detailed in the next chapter.

This chapter can safely be skipped by those who feel thoroughly at home with the nature, requirements and methods of multi-user programming. All others should read this chapter carefully.

WHAT IS MULTI-USER PROGRAMMING

Multi-user programming is the writing of programs that are designed to run in an environment where more than one program can execute at the same time, on the same computer, using the same resources. Such an environment is termed a multi-user environment.

In a single-user environment, a user has an entire computer to himself. Under such conditions, a program may safely assume:

1. Constancy: The data on file in the computer does not change during program execution unless the program explicitly changes it.
2. Consistency: The data on file is consistant unless an error has occurred.
3. Availability: Any device of the computer is available for immediate use at any time.

In a multi-user environment, two or more users share a computer and its resources. They appear to use it simultaneously, but that is an illusion. The computer actually works on only one user's program at a time, but it cycles from program to program so rapidly that all the programs seem to be executing at once.

In a multi-user environment, the program a computer is working on at a given moment is termed the active program; all other programs are said to be suspended. An active program can never predict when or for how long it will be suspended, and a program returned to active status cannot determine that any suspension has occurred.

The assumptions of constancy, consistency, and availability listed above do not hold in a multi-user environment. Specifically:

1. A program cannot assume that any data accessible to any other program is constant. Some other program may change the data while the program is suspended, and the program can thereby be left with wrong data values that cause it to fail.
2. A program cannot assume that any data accessible to any other program is consistant. The data it sees may be inconsistant because some other program began modifying it, but was suspended before the modification could be completed.
3. A program cannot assume that every system device is available to it for use at any time. Some or all of the devices may be in use by other programs.

Put more generally, a program executing in a multi-user environment cannot behave as though it were in a single-user environment. It must assume that other programs exist as well, and take specific steps to avoid interfering with them or being interfered with by them. If it does not do so, disastrous program and system failures can result.

CONCURRENCY

Whenever two or more programs share a computer and its resources, they are said to execute concurrently. The sharing may be direct, as when several timesharing users share a computer, or indirect, as when network users share the resources of the network.

Whenever programs are executing concurrently that use the same system resources, there is the possibility that they will try to use the same resource at the same time. If this occurs, neither will be able to use the resource effectively, because neither will have full control of it. This is the mechanism whereby a program's erroneous assumption that it is alone in its computer can cause trouble.

The special problems concurrent programs can encounter because of each-other's effects on shared computer resources are called concurrency problems.

CONCURRENCY PROBLEMS

There are three basic concurrency problems of interest here: conflicting update, inconsistant data, and uncoordinated device access. The following sections define and give examples of each of these problems.

Conflicting Update

Conflicting update can occur when a concurrent program erroneously assumes constant data.

Suppose that two inventory programs both need to modify the quantity of some part in stock. The program INPRT wants to record that ten more of PART1 have been received. Program OUTPRT wants to record that five PART1s have been sent to manufacturing. If there are 20 PART1s on record before the two programs execute, there should be 25 afterwards.

Both INPRT and OUTPRT need to perform the same three actions:

1. Read a quantity record from a file
2. Modify the quantity in the record
3. Write the modified record back to the file

Suppose that INPRT and OUTPRT are run consecutively, so that INPRT finishes before OUTPRT begins. The sequence then is:

Program	Action	Quantity in Program	Quantity in File
INPRT	READ FILE	20	20
INPRT	ADD 10 TO QUANTITY	30	20
INPRT	WRITE FILE	30	30
OUTPRT	READ FILE	30	30
OUTPRT	SUB 5 FROM QUANTITY	25	30
OUTPRT	WRITE FILE	25	25

The result is 25, which is correct.

If the programs execute concurrently, however, each can change the system while the other is suspended, causing false data to be created. Consider the sequence:

Line	Program	Action	Quantity in Program	Quantity in File
1	INPRT	READ FILE	20	20
2	OUTPRT	READ FILE	20	20
3	INPRT	ADD 10 TO QUANTITY	30	20
4	INPRT	WRITE FILE	30	30
5	OUTPRT	SUB 5 FROM QUANTITY	15	30
6	OUTPRT	WRITE TO FILE	15	15

The result is 15, which is incorrect.

The error occurs in Line 5. OUTPRT is assuming that the system's data is constant, so that the value it read from the file in Line 2 must still be in the file when it executes Line 5. But the value has changed, because INPRT modified it in Lines 3 and 4 while OUTPRT was suspended. The result is an erroneous value for the quantity of PART1.

Inconsistant Update

Inconsistant update can occur when a concurrent program erroneously assumes consistant data.

Consider a system with two files. One contains master data records and is called the Data File. The other is an index into the first file, used for rapid data retrieval, and is called the Index File.

Suppose there are two file-access programs operating on these two files. NEWREC wants to add a new record to the Data File, then update the Index File to reflect the new record. PRTREC wants to use the Index File to find a record in the Data File, then print the data in the record it has found.

Suppose that NEWREC and PRTREC are run consecutively so that NEWREC finishes before PRTREC begins. The sequence is:

Program	Action
NEWREC	Access Data File
NEWREC	Access Index File
NEWREC	Modify data
NEWREC	Modify index(es)
NEWREC	Write Data File
NEWREC	Write Index File
PRTREC	Access Index File
PRTREC	Access Data File
PRTREC	Print data from Data File

This sequence results in no difficulty.

If the programs execute concurrently, however, PRTREC can encounter inconsistant data, because it may become active while NEWREC is in suspension. Consider the sequence:

Line	Program	Action
1	NEWREC	Access Data File
2	NEWREC	Access Index File
3	NEWREC	Modify data
4	NEWREC	Modify index(es)
5	NEWREC	Write Data File
6	PRTREC	Access Index File
7	PRTREC	Access Data File
8	PRTREC	Print data from Data File
9	NEWREC	Write Index File

Here the error is in Line 7. PRTREC is assuming that the system's data is consistent, so that the data it reads from the Index File correctly describes the Data File. But no such consistency exists, because NEWREC was suspended after it wrote the modified data file, but before it could write the correspondingly modified index file. PRTREC therefore encounters inconsistant data when it attempts Line 7, and cannot be relied on to execute correctly thereafter.

Uncoordinated Device Access

Uncoordinated device access can occur when a concurrent program assumes erroneously that it has the computer's devices all to itself.

Suppose two programs, PRINT1 and PRINT2, each want to print a file on the line printer. If they run consecutively, all will be well; both files will be printed successfully. But if they run concurrently, each program writing to the printer independently of the other, the output on the printer will be a random jumble of the contents of the two files.

A similar problem can occur when a disk drive is accessed. The basic steps for accessing a floppy disk are:

1. Move the head to the desired track
2. Lower the head
3. Read/write the track
4. Raise the head

Suppose PROG1 and PROG2 each want to read a disk track. If they execute sequentially, they will not interfere with each other. If they execute concurrently, a sequence could result such as:

Line	Program	Action
1	PROG1	Move head to track
2	PROG2	Move head to track
3	PROG1	Lower the head
4	PROG1	Read the track
5	PROG1	Raise the head
6	PROG2	Lower the head
7	PROG2	Read the track
8	PROG2	Raise the head

The track PROG1 reads in Line 4 is not the track it selected in Line 1, but the track PROG2 selected in Line 2. The consequences are sure to be unpleasant.

PREVENTING CONCURRENCY PROBLEMS

All the concurrency problems just described have the same cause: concurrent programs, each using the system as if no other program existed, trying to use the same resource at the same time. All can be prevented in the same way by providing a mechanism whereby the concurrent programs can detect and respond appropriately to each other' s resource usages.

Resource Locks

The required mechanism is called a resource lock. A resource lock has four basic properties:

1. It can be set by a program on a system resource.
2. When set on a resource by a program, the lock makes the resource unavailable to all other programs.
3. Once set, a lock remains set even though the program that set it is suspended.
4. A lock can be cleared (unlocked) by the program that set it, freeing the resource for use by other programs.

When resource locks are used, a program that wants to perform an action that could impair or be impaired by concurrently executing programs does the following:

1. Lock all the resources needed for the actions.
2. Perform the action.
3. Unlock the resources.

Resource locks prevent concurrency problems by allowing a program to deny other programs access to the resources the program is using. In effect, they allow a concurrently executing program to create a single-user environment for itself whenever it needs the constancy, consistancy, and accessibility that only such an environment can provide.

When a program tries to lock a resource that is already locked, indicating that some other program is using it, the program is suspended, and remains suspended until the resource becomes unlocked. The program is then activated, and may lock the resource and proceed. If several programs try to lock a locked resource, their requests are queued.

The processes of suspending, queueing, and activating programs that attempt to set locks are handled by the operating system. Programs need be concerned only with setting and releasing locks; the system does the rest automatically.

EXAMPLES OF RESOURCE LOCK USAGE

The following sections continue the examples given previously for conflicting update, inconsistant update, and uncoordinated device access. For each example, the way in which resource locks can prevent the applicable concurrency problem is illustrated. Each illustration is, of course, only one possibility among many, but in each case all the possibilities are equivalent with respect to the concurrency problem under discussion.

Resource Locks and Conflicting Update

INPRT wants to add 10 to a quantity. OUTPRT wants to subtract 5 from the quantity. The quantity is initially 20. With resource locks, the sequence could be:

Program	Quantity in Action	Quantity in Program	Quantity in File
INPRT	Lock file	-	20
INPRT	Read file	20	20
OUTPRT	Try to lock file	-	20
OUTPRT	Suspended until file free	-	20
INPRT	Add 10 to quantity	30	20
INPRT	Write file	30	30
INPRT	Unlock file	30	30
OUTPRT	Become activated	-	30
OUTPRT	Lock file	-	30
OUTPRT	Read file	30	30
OUTPRT	Subtract 5 from quantity	25	30
OUTPRT	Write file	25	25
OUTPRT	Unlock file	25	25

The result is 25, which is correct.

Resource Locks and Inconsistant Update

NEWREC wants to add a record to a Data File, then update an Index File accordingly. PRTREC wants to use the Index File to find and print a record from the Data File. With resource locks, one possible sequence is:

Program	Action
NEWREC	Lock Data File
NEWREC	Lock Index File
NEWREC	Read Data File
NEWREC	Read Index File
NEWREC	Modify data
NEWREC	Modify indexes
NEWREC	Write Data File
PRTREC	Try to lock Index File
PRTREC	Suspended until Index File free
NEWREC	Write Index File
NEWREC	Unlock Data File
NEWREC	Unlock Index File
PRTREC	Become activated
PRTREC	Lock Index File
PRTREC	Lock Data File
PRTREC	Access Index File
PRTREC	Access Data File
PRTREC	Print data from Data File
PRTREC	Unlock Data File
PRTREC	Unlock Index File

No problem results from this sequence.

Resource Locks and Device Access

To prevent the problem of concurrent printer use causing jumbled output, a program that needs to use a printer need only lock the printer before starting to print, and unlock it when the printing is done.

PROG1 and PROG2 both want to read a disk track. With resource locks the sequence could be:

Program	Action
PROG1	Lock disk drive
PROG1	Move head to track
PROG2	Try to lock disk drive
PROG2	Suspended until disk drive free
PROG1	Lower the head
PROG1	Read the track
PROG1	Raise the head
PROG1	Unlock the drive

Program	Action
PROG2	Become activated
PROG2	Lock the drive
PROG2	Move head to track
PROG2	Lower the head
PROG2	Read the track
PROG2	Raise the head
PROG2	Unlock the drive

No problem results from this sequence.

Most operating systems, including OS-65U, automatically handle concurrent device access on behalf of all programs. With such an operating system, programs do not need to take explicit actions to prevent concurrency problems when accessing a device. They need only request the devices they want; the operating system will apply resource locks as needed.

IMPLEMENTING RESOURCE LOCKS

There are many ways to implement resource locks. OSI computers implement them through the use of semaphores.

Semaphores

A semaphore is a special flag provided by the operating system. A semaphore has two states: set (1) and clear (0). Only two operations are possible on a semaphore: to set it and to clear it.

All aspects of semaphore maintenance are handled by the operating system. A program cannot set or clear a semaphore directly; it can only give a set-semaphore or clear-semaphore request to the operating system. The operating system handles such requests as follows:

- When a program asks the operating system to set a semaphore that is clear, the system sets the semaphore, and the program continues execution.
- When a program asks the operating system to set a semaphore that is already set, the system suspends the program. The program will remain suspended until the semaphore is cleared.
- When a program asks the operating system to clear a semaphore that is set, the system clears the semaphore. If there is a suspended program waiting for the semaphore to clear, the system activates the program and sets the semaphore for it.
- When more than one program attempts to set a set semaphore, their requests are queued. Each time the semaphore clears, the program that has waited longest is activated.

Once a program has asked the operating system to set a sempahore, that program cannot be suspended in favor of another program until the set request has been completely processed. Without this provision, programs trying to set the same semaphore could encounter the conflicting update problem with respect to the state of the semaphore.

A program that has been suspended until a semaphore clears is said to be waiting on that semaphore.

Locking with Semaphores

The first step in using semaphores to lock resources is to assign a semaphore to every resource that can be used concurrently. This assignment is purely a convention of usage; the computer itself does not know or care what semaphore has been assigned to what resource.

To lock a resource, a program simply asks the operating system to set the resource's semaphore.

If the semaphore is clear at the time of the request, the system sets it. The program can now be certain that it has sole access to the resource: if the resource had been in use by any other program, its semaphore would have been set already.

If the semaphore is set at the time of the request, the system suspends the program until the semaphore is clear. When the semaphore clears, the system activates the program and sets the semaphore. The program can again be sure it has sole access to the resource: if the resource were still in use by any other program, the requesting program would not have been activated.

To unlock a resource, a program asks the operating system to clear the resource's semaphore. The system always carries out this request immediately. If a program was waiting on the semaphore, that program is activated as described above. If several programs are waiting, the one that has been waiting longest is activated. If no program is waiting, the program that unlocked the resource continues execution.

More generally, when a program needs sole access to a resource, it brackets the code that performs the access between a set-semaphore request and a clear-semaphore request. The program can then be certain that the operating system will permit it to execute the code between the brackets only when no other program is accessing the resource.

Locking Multiple Resources

For simplicity, the preceding discussion assumed that a program locks only one resource at a time. In practice, a program will often need to have several resources locked at the same time in order to execute without interference.

When each of a group of concurrently executing programs needs at most one resource at a time, concurrency problems can be prevented by using semaphores as just described. But when two or more concurrent programs each need to lock more than one resource at a time, a new concurrency problem becomes possible: deadlock.

DEADLOCK

Deadlock occurs when two or more concurrent programs are suspended because each is waiting for a resource that has been locked by one of the others. None of the programs will ever be activated, because they are all waiting for each other.

Consider the example used previously for inconsistant data, in which NEWREC wants to add to a Data/File and update its Index File, while PRTREC wants to use the Index File to find and print a record from the Data File. The sequence shown above, in which these programs used resource locking to avoid inconsistant data, encountered no problems. Other equally likely sequences would not fare so well, for example:

Program	Action
NEWREC	Lock Data File
PRTREC	Lock Index File
NEWREC	Try to lock Index File
NEWREC	Suspended until index file free
PRTREC	Try to lock Data File
PRTREC	Suspended until Data File free

But the Data File will never be free, because the only program that could free it is suspended waiting for the Index File; and the Index File will never be free, because the only program that could free it is suspended waiting for the Data File. This is deadlock.

Preventing Deadlock

Deadlock can be prevented entirely if all concurrently executing programs lock the resources they need in the same order. This order, like the assignment of semaphores to resources, is purely a convention. All that matters is that the convention be followed by every concurrent program.

With ordered resource locking, the sequence above could become:

Program	Action
NEWREC	Lock Data File
PRTREC	Try to lock Data File
PRTREC	Suspended until Data File free
NEWREC	Lock Index File
NEWREC	Process Data and Index Files
NEWREC	Unlock Data File
PRTREC	Activated because Data File free
PRTREC	Lock Data File
PRTREC	Try to lock Index File
PRTREC	Suspended until Index File free
NEWREC	Unlock Index File
PRTREC	Activated because Index File free
PRTREC	Lock Index File
PRTREC	Process Data and Index Files
PRTREC	Unlock Data File
PRTREC	Unlock Index File

The order in which resources are unlocked is immaterial to deadlock prevention. However, a slight increase in system throughput can be gained by unlocking resources in the same order in which they are locked.

PROGRAMMING IN THE OS-65U MULTI-USER ENVIRONMENT

This chapter assumes that you are thoroughly familiar with the material covered in the previous chapter. If you are not, be sure to read and understand that chapter before proceeding. The terms and concepts used in this chapter are all defined in that one, and none of those definitions are repeated here.

This chapter discusses:

- OSI System and User Semaphores
- Using OSI Semaphores
- Timed Semaphore Waits
- Locking File Records
- Running Multi-User Programs in a Single-User Environment
- Concurrency and Printer Control
- Concurrency and System Utilities
- Semaphore Errors
- Operator Intervention
- The SEMCHK Utility

OSI SEMAPHORES

An OSI computer offers two types of semaphores for use in preventing concurrency problems:

- System semaphore
- User semaphore

System Semaphores

In a multi-user environment, many copies of the operating system may be operating concurrently. These copies must co-ordinate their activities just as user programs must. They accomplish this co-ordination by using system semaphores.

There are 56 system semaphores in every OSI computer; semaphore 0, and semaphores 201 to 255. Whenever a copy of OS-65U needs exclusive access to a system resource, it locks the resource by setting the appropriate semaphore. In OS-65U Version 1.44, the system semaphores are allocated as follows:

Number(s)	Usage
0	Reserved for CP/M Network Extension
201 - 206	Devices A-F
207 - 210	Reserved
211 - 219	Logical Device locks for CP/M Network Extension
220 - 230	Print Spooling
231 - 244	Reserved
245 - 251	Output Peripheral Devices 3-9
252 - 253	Reserved
254	Hard Disk Controller
255	Floppy Disk Controller

System semaphores cannot be set or cleared directly by a user or program. They are set and cleared as needed by OS-65U, and their correct operation can be assumed when commands and programs are executed.

User Semaphores

Most of the concurrency problems that could occur in the OSI multi-user environment are prevented automatically by OS-65U through the use of system semaphores. There is one type of problem that cannot be handled by OS-65U: concurrent access to data files. Prevention of problems due to concurrent file access is the responsibility of the programs accessing the files.

Every OSI computer contains 200 user semaphores, numbered 1-200, for use in co-ordinating concurrent data access. Whenever an executing program needs exclusive access to a file, it locks the file by setting a semaphore associated with it.

The question of which semaphore corresponds to which file is entirely a matter of local convention. Each OSI computer site must determine for itself what semaphore is to lock what file, and must take care that every program that uses a shared file first locks it by setting the semaphore assigned to it.

As discussed in the last chapter, a program that needs several resources at the same time must lock them in a standard order, or deadlock may occur. The order used is also a matter of local convention: all that matters is that the convention be followed by every concurrent program. One convenient method is to lock resources in order by semaphore number.

USING OSI SEMAPHORES

Only two BASIC commands are necessary to use semaphores on an OSI computer: one to set a semaphore, and one to clear it.

The command to set a semaphore is

```
WAIT FOR n
```

where n is a user semaphore number. The n may be any numeric expression. If n is not integral, it will be truncated to an integer. The value of n after any evaluation or truncation must be between 1 and 200 inclusive.

The command to clear a semaphore is:

```
WAIT CLEAR n
```

where n is the number of a semaphore previously set by the program with a WAIT FOR command.

All semaphores set by a user are automatically cleared when the immediate mode of BASIC is entered.

Semaphores and Keyring Networks

In a keyring network, a user may access resources in up to three different computers. Each of these computers has its own array of semaphores, and each set uses the same numbers. Therefore, a program that wants to set or clear a semaphore while networking is in progress must be able to specify which computer the desired semaphore is on.

A simple rule takes care of this need: whenever OS-65U processes a WAIT FOR or WAIT CLEAR command, the semaphore it accesses will be on the computer that holds the device specified by the user's most recently executed DEV command. If no DEV commands have been given, the user remains by default on the local device from which he was booted; a semaphore from his own computer would therefore be used.

Timed Semaphore Waits

In the default case, a program waiting on a semaphore will wait indefinitely; only the clearing of the semaphore can reactivate it.

It is possible to specify a time limit on a WAIT FOR command. When such a time limit has been specified, a program waiting on a semaphore will wait only until the time limit has expired. The program will then be reactivated even though the semaphore it was waiting on has not been cleared.

To establish a time limit for semaphore waits, execute:

```
POKE 19632,s
```

where s is the number of seconds to wait, from 0 to 59.

If 1-59 is specified, the program will wait the corresponding number of seconds before being reactivated by the operating system.

If 0 is specified, the program will not wait at all if the semaphore is already set; it will just continue executing.

Once a time limit has been established, it applies to all subsequent semaphore waits. The time limit can be changed as often as desired by executing additional POKEs. To return to the default of indefinite waiting, execute:

```
POKE 19632,60
```

A program reactivated after a timed semaphore wait cannot just proceed to use the resource for which it was waiting. First, it must determine whether it has become active because it has successfully locked the resource, or because timeout occurred on the wait. It determines this by PEEKing location 19633:

```
SS = PEEK(19633)
```

If the value at 19633 is nonzero, the program has successfully locked the resource and may proceed to use it. If the value is 0, the resource is still in use by some other program and therefore must not be used by the program that was waiting for it.

The following is a simple example of a timed wait. It shows one way a program can respond when timeout occurs on a wait.

```
 100   POKE 19632,25: REM WAIT 25 SECONDS
 200   WAIT FOR 1
 300   IF PEEK (19633) = 0 GO TO 1000: REM TIMEOUT
 400   REM CODE USING THE RESOURCE APPEARS HERE
 500   WAIT CLEAR 1
 600   GO TO 2000
1000   PRINT "TIMEOUT ON SEMAPHORE WAIT."
1100   PRINT "RETRY THE WAIT? (Y/N):"
1200   INPUT A$
1300   IF A$ = "Y" GO TO 200
2000   POKE 19632,60: REM RETURN TO INDEFINITE WAITING
3000   REM ETC.
```

Timed Semaphores Waits and Multiple Resource Usage

Timed waits can be used to increase system throughput in cases where programs lock large numbers of resources at the same time. This is one of their most useful applications.

Suppose that a program must lock many resources before using any of them. If it can get all of them fairly quickly, no problem arises. But if it gets some of them, then encounters an extremely long wait, all the resources it has locked will be unavailable to other programs for the duration of the wait, even though they are doing the program that has locked them no good. The result can be a great deal of needless waiting, and a corresponding reduction in system throughput.

This problem would not occur if a program that encountered a long wait partway through a resource locking sequence relinquished the resources it had already locked, and then retried the whole acquisition sequence from the beginning.

The following program shows how to use timed waits to prevent throughput degradation due to unnecessary writing.

```
100     REM PUT THE SEMAPHORE NUMBERS OF THE
200     REM RESOURCES TO BE LOCKED IN AN ARRAY
300     RN(1)=2: RN(2)=5: RN(3)=13: RN(4)=25: RN(5)=112
400     POKE 19632,30: REM SET TIME LIMIT TO 30
500     FOR L = 1 TO 10: REM MAX 10 TO AVOID INFINITE LOOP
600     FOR I =1 TO 5: REM LOOP TO LOCK RESOURCES
700     WAIT FOR RN(I): REM LOCK EACH RESOURCE IN TURN
800     IF PEEK (19633) = 0 GO TO 1200: REM TIMEOUT OCCURRED
900     NEXT I
1000    REM IF HERE, ALL FIVE RESOURCES HAVE BEEN LOCKED
1100    GO TO 1000: REM JUMP TO CODE THAT USES THE RESOURCES
1200    REM UNLOCK RESOURCES WHEN A WAIT HAS LASTED TOO LONG
1300    IF I = 1 GO TO 1600: REM POSSIBLY NO RESOURCES WERE LOCKED
1400    FOR J = 1 TO I-1 REM LOOP TO UNLOCK RESOURCE
1500    WAIT CLEAR RN(J): REM UNLOCK EACH RESOURCE IN TURN
1600    NEXT J
1700    NEXT L
10000   REM CODE HERE TO USE RESOURCES
11000   REM CODE HERE TO UNLOCK RESOURCES
12000   END
```

Locking a File Record

It is not always acceptable to require that a whole file be locked before any part of it can be used by a concurrent program. In the case of a large data file needed by many programs, such a requirement would drastically reduce system throughput.

The solution is to lock individual records rather than whole files. This is done by putting a usage flag in each record. A usage flag is a field to which a 1 or 0 can be assigned, to indicate whether the record is or is not in use. If all programs correctly set, clear, and respect these usage flags, they become resource locks.

The general steps for locking, processing and unlocking a record are as follows:

1. Lock the whole file
2. Read the desired record's lock field
3. If the record lock is set, unlock the file and go to 1.
4. If the record lock is clear, set it.
5. Write the record lock field to the file.
6. Unlock the file.
7. Read and process the locked record's data fields as needed. Clear the lock field.

8. Lock the file.

9. Write the cleared lock field and any modified data fields to the file.

10. Unlock the file.

The following program illustrates the use of record locking under OS-65U. This program assumes familiarity with Chapter 4, "File Organization and Procedures". The record lock field used in the program is two characters long: the first character holds a "0" or "1", while the second holds the <cr> used by OS-65U to separate file fields. The record lock field is assumed to be the first field in the record, and to be followed by three data fields called A$, B$, and C$.

```
100   REM F$ = NAME OF FILE
200   REM P$ = PASSWORD OF FILE
300   REM LF = INDEX FOR LOCK FIELD OF DESIRED RECORD
400   REM L$ = VARIABLE FOR RECORD LOCK FIELD DATA
1000  WAIT FOR 15: REM LOCK THE FILE
1100  OPEN F$, P$, 1
1200  INDEX <1> = LF: REM SET INDEX TO READ DESIRED LOCK  FIELD
1300  INPUT%1, L$: REM READ THE LOCK FIELD
1400  IF L$="0" GO TO 1900: REM TEST LOCK FIELD
1500  REM CODE FOR LOCKED RECORD (L$="1")
1600  CLOSE 1: REM MUST CLOSE-OPEN TO FORCE RE-READING OF LOCK FIELD
1700  WAIT CLEAR 15: REM UNLOCK THE FILE
1800  GO TO 1000: REM RETRY UNTIL RECORD BECOMES UNLOCKED
1900  REM CODE FOR UNLOCKED RECORD (L$="0")
2000  INDEX <1> = LF: REM RESET INDEX TO VALUE FOR LOCK FIELD
2100  L$ = "1": REM SET FIELD VARIABLE
2200  REM WRITE LOCK FIELD.  CLOSE FILE TO FORCE BUFFER DUMP.
2300  PRINT%1, L$: CLOSE 1
2400  WAIT CLEAR 15: REM UNLOCK THE FILE
2500  OPEN F$, P$, 1: REM OPEN FILE FOR PROCESSING
2600  L$ = "0": REM CLEAR LOCK FIELD VARIABLE
2700  REM READ AND PROCESS DATA FIELDS AS NEEDED
2800  REM FOR THIS EXAMPLE ASSUME THAT VARIABLES
2900  REM A$, B$ AND C$ ARE READ AND MODIFIED
3000  REM NOW WRITE CLEARED LOCK FIELD AND NEW DATA FIELDS
3100  WAIT FOR 15: REM LOCK THE FILE
3200  CLOSE 1: OPEN F$,P$,1: REM SEE TEXT FOLLOWING THIS EXAMPLE
3400  INDEX <1> = LF: REM RESET TO VALUE FOR LOCK FIELD
3500  REM WRITE ALL FIELDS.  CLOSE TO FORCE BUFFER DUMP
3600  PRINT%1, L$,A$,B$,C$: CLOSE 1
3700  WAIT CLEAR 15: REM UNLOCK THE FILE
```

CAUTION

Line 3200 in the above program exists to prevent a subtle and very dangerous problem that can arise when record locking is used. Be sure to read the rest of this section carefully if you plan to use record locking.

Two granularities characterize every file in a computer system. One is the granularity imposed by the file's record structure.This is the granularity a program sees when it processes a file. The other granularity is imposed by the I/O handlers that read and write blocks of data stored on disk. This granularity cannot be directly detected by a program, but it is nevertheless capable of causing trouble to programs executing in a multi-user environment. Understanding the problem that can arise requires a brief discussion of disk I/O.

A disk I/O always reads or writes at least one whole data block. The block is read into or written from an I/O buffer that is maintained by OS-65U for use in accessing disk data. There is a separate I/O buffer for every file opened by a program. Different programs accessing the same file use different I/O buffers.

The actions performed by OS-65U to process a file-directed INPUT statement are:

I1. Read the disk block containing the record requested by the INPUT statement into the program's I/O buffer for the file.

I2. Move data from the buffer to the program variables listed in the INPUT statement.

The actions performed by OS-65U to process a file-directed PRINT statement are:

P1. If it is not already there, read the disk block containing the record that contains the data to be PRINTed into the program's I/O buffer for the file.

P2. Move data from the program variables listed in the PRINT statement to the I/O buffer.

P3. Write the modified disk block from the I/O buffer back to the file.

A problem can arise because the same disk block can contain more than one record. Different programs locking and modifying different records in the same block can therefore encounter the problem of conflicting update even though they use record locks correctly.

Suppose the two concurrent programs, PROG1 and PROG2, each need to modify a file record. PROG1 needs to modify REC1 while PROG2 needs to modify REC2. REC1 and REC2 are both in the same disk block, so that reading or writing either one necessarily reads or writes the other as well. The following abbreviated sequence shows how PROG1 and PROG2 could get into trouble:

Program	Action
PROG1	Lock REC1
PROG2	Lock REC2
PROG1	Read REC1. REC2 is also read
PROG2	Read REC2. REC1 is also read
PROG1	Modify REC1
PROG2	Modify REC2

Program	Action
PROG1	Write modified REC1. REC2, which has remained in THE PROG1 I/O buffer, is also written.
PROG2	Write modified REC2. REC1 is also written.

The problem occurs in the last line. The copy of REC1 in PROG2's buffer has become obsolete due to the action of PROG1. When PROG2 writes REC2, obsolete copy of REC1 is written also, and the modification of REC1 made by PROG1 is thereby undone. This is a typical case of conflicting update.

This problem would be easy to prevent if disk blocks could be locked, but that is not possible under OS-65U. The other solution is to have every program that writes to a lockable record update its I/O buffer before performing the write. This is done by closing and opening the file after it is locked and before anything is written to it, as shown above in Line 3200.

This method works because closing and opening a file causes its I/O buffer to be cleared. Then, when OS-65U processes a PRINT statement for the file, it must begin by reading the block containing the data to be PRINTed (Step P1, above). Since this read occurs after the whole file has been locked, it is guaranteed to place an up-to-date copy of the block in the file's I/O buffer. Conflicting update is thereby provided.

With close-open used as described, the sequence above could become as follows. Only those closings and openings that bear on the point of the example are shown.

Program	Action
PROG1	Lock REC1.
PROG2	Lock REC2.
PROG1	Read REC1. REC2 is also read.
PROG2	Read REC2. REC1 is also read.
PROG1	Modify REC1.
PROG2	Modify REC2.
PROG1	Lock, close and open file.
PROG1	Write modified REC1. The block that contains REC1 must first be reread from disk.
PROG1	Unlock file.

Program	Action
PROG2	Lock, close and open file.
PROG2	Write modified REC2. The block that contains REC2 must first be reread from disk. PROG2's buffer thereafter contains the latest version of REC1, and this version will be preserved when REC2 is written.
PROG2	Unlock file.

Locking Multiple Records

When concurrent programs need to have more than one file record locked at the same time, deadlock becomes possible, as with any other group of resources.

When records from more than one file must be locked simultaneously, the files must be processed in the same standard order that would be used if file locking rather than record locking were being done.

Where more than one record from a file must be locked simultaneously, the individual records must be locked in a standard order. The order is a matter of local convention. One convenient method is to lock the records in the sequence in which they exist in the file. Different files can use different standard orders, so long as the same order is always used on anyone file.

For example, a sequence for locking several records from each of several files could be:

```
Lock File 6
Lock File 6 Record 7
Lock File 6 Record 11
Unlock File 6
Lock file 9
Lock File 9 Record 8
Lock File 9 Record 42
Unlock File 9
Lock File 42
.
.
.
```

RUNNING MULTI-USER PROGRAMS IN SINGLE-USER MODE

When a program written for a multi-user environment is run in single user mode, any WAIT FOR and WAIT CLEAR commands in it are ignored by OS-65U. The existence of WAIT commands in a program therefore generates no problems when the program is run in single-user mode.

A multi-user program that specifies a timed semaphore wait cannot, however, execute without modification in single-user mode. No program executing in single-user mode may specify a timed wait. Therefore, any program that uses a timed wait must be converted before it can run in single-user mode.

One way to convert such a program is to eliminate the line(s) that specify the waits. With those lines deleted, the problem disappears. The disadvantage of this method is that the converted program can no longer execute as designed in multi-user mode.

A more flexible conversion can be performed by making execution of the line(s) that specify timed waits conditional on the mode in which the program is running.

The mode in which a program is running can be determined by PEEKing location 16317. If the value there is greater than 1, the program is not running in single-user mode. The conversion is performed by prefixing

```
IF PEEK (16317) > 1 THEN
```

to every POKE statement that establishes or eliminates timed waiting.

For example, the program given previously as a simple example of a timed wait could run in single-user mode if it were converted to the following. The only changes are in Line 100 and Line 2000.

```
  99    REM CONDITIONALLY ESTABLISH TIMED WAIT
 100    IF PEEK (16317) <> 1 THEN POKE 19632,25
 200    WAIT FOR 1
 300    IF PEEK (19633) = 0 GO TO 1000: REM TIMEOUT
 400    REM CODE USING THE RESOURCE APPEARS HERE
 500    WAIT CLEAR 1
 600    GO TO 2000
1000    PRINT "TIMEOUT ON SEMAPHORE WAIT"
1100    PRINT "RETRY THE WAIT? (Y/N):"
1200    INPUT A$
1300    IF A$ = "Y" GO TO 200
1999    REM CONDITIONALLY RETURN TO INDEFINITE WAITING
2000    IF PEEK (16317) <> 1 THEN POKE 19632,60
3000    REM ETC.
```

CONCURRENCY AND PRINTER CONTROL

An OSI computer has three devices that can be used to transmit data to a printer. They are:

3: Serial I/O Port
5: Parallel Output Port (Centronix)
8: Serial I/O Port

The system utility WPDRIV (see Chapter 8) can be used to create a fourth printer device:

6: Parallel Output Port (Diablo)

Devices 3 and 8 can be used to input as well as to print, but that does not matter here because no concurrency problems can arise when these devices are used for input. In the following, devices 3, 5, 6, and 8 are referred to generically as printers.

Printers are operating system resources, and as such are managed by OS-65U using system semaphores. These semaphores, like all system semaphores, may not be directly accessed by users. Instead, OS-65U manages its printers automatically on behalf of all users, using information derived from each user's PRINT and FLAG commands. The following are the rules by which OS-65U coordinates printer usage:

1. When a user executes a PRINT directed to a printer not currently in use, that printer is locked on behalf of the user. Only the user may thereafter use the printer, until it is unlocked.

2. When a user issues a PRINT directed to a printer that is currently locked for someone else, the user is suspended until the printer becomes free. If the PRINT was issued from a program, the program is suspended. If the PRINT was issued for immediate mode, the user's terminal handler is suspended. His terminal will then be unresponsive until the printer becomes free and the PRINT command has executed.

3. When a user issues a FLAG 100 or a FLAG 101 (the Device 5 Top of Form commands) the system responds as if he had issued a PRINT command directed at Device 5.

4. Whenever a user enters immediate mode, any printers locked for him are unlocked.

5. When a user issues the command:

   PRINT #n!

   where n is a printer device number, the printer denoted by n is unlocked.

6. Whenever a printer is unlocked, OS-65U issues a top-of-form command to it before permitting it to be locked again.

Frequently a concurrent program will need to use a printer while it has other resources locked as well. Sometimes a program will need to have more than one printer locked at the same time. As always when more than one resource is locked simultaneously, deadlock is possible and must be prevented through use of a standard locking order. One convenient method is to always lock printers last, after all file and record locks have been set, and where more than one printer is needed at a time, to lock them in order by device number.

CAUTION

The system utility PRTMAP (see Chapter 8) can be used to change printer device mapping. For example, PRTMAP can be used to set the system so that all output directed to Device 5 by a program is redirected to Device 8. However, the locking and unlocking of printers does not change when PRTMAP has been used. Therefore, if PRTMAP has redirected Device 5 output to Device 8, and one program prints to Device 5 while another concurrently prints to Device 8, the outputs of the two programs will be jumbled together on Device 8. The solution to this problem is either to avoid it in the first place, or to allocate a user semaphore for coordinating the device on which the problem is occurring.

CONCURRENCY AND SYSTEM UTILITIES

OS-65U provides various system utilities which operate on shared resources such as system directories. All the usual concurrency problems are possible when these utilities are run in a multi-user enviornment.

OS-65U prevents system utility concurrency problems in two ways:

1. It uses system semaphores to protect shared resources used by system utilities.

2. It modifies the actions of some system utilities, so that they cannot perform certain actions that are useful in a single-user environment, but cannot be executed safely in a multi-user environment.

As a result, some system utilities behave differently in multi-user mode than they do in a single-user mode. The utilities that change, and the nature of those changes, are as follows:

- The CREATE, RENAME and DELETE programs can be used by only one user at a time on a given disk. A second user who attempts to use one of these programs on a disk will receive a message asking him to try again later.

- COPYFI can be used by only one user at a time on a given destination file. A second user who tries to copy to that file will receive a try-again-later message.

- COPIER can be used only to copy the files portion of a disk. Copying a disk's system portion can be done only in single-user mode.

- PACKER, CHANGE and LOAD48 cannot be used concurrently at all. They can be run only in single-user mode.

All other system utilities behave the same way in a multi-user environment that they do in a single-user environment.

SEMAPHORE ERRORS

A program may have at most 16 semaphores set at any one time. An attempt to set a 17th semaphore will result in an SS (Semaphore Stack Overflow) error, and the semaphore will retain the state it was in before the attempt.

If a program tries to clear a semaphore that is already clear, an FC error occurs if the program is running on a timesharing partition, and Error 239 occurs if the program is running on a workstation.

If a program tries to clear a semaphore that was set by some other program, an FC error occurs. The semaphore will nevertheless be cleared. This can have harmful effects on the program that set the semaphore, and possibly on other programs as well.

If a program tries to set a semaphore already set by itself, it will be suspended indefinitely. It is waiting for itself to become active and clear the semaphore on which it is waiting, a wait that will never end without outside intervention. One could think of this as autodeadlock.

OPERATOR INTERVENTION

There are some cases in which operator intervention is necessary to keep a multi-user environment running smoothly. The most common cause is a program that has hung up after locking one or more system resources. Another common cause is that concurrent programs have tried to lock resources in an inconsistant order, and have consequently gone into deadlock. In either case, operator intervention is required to free the locked resources. Two steps are required:

1. Cancel any hung up or deadlocked programs.
2. Clear any semaphores that have been left set. Such semaphores are hereafter called stuck semaphores.

To cancel a hung or deadlocked program on a workstation, reboot the workstation. This also clears any semaphores local to the workstation. To cancel a program on a timesharing partition, restart the partition. The computer's semaphores will be unaffected by this restart.

There are two ways to clear a stuck semaphore: from immediate mode, or with the SEMCHK utility. Only user semaphores can be cleared from immediate mode. Any semaphore can be cleared with SEMCHK.

To clear a user semaphore from Immediate Mode, enter that mode, then execute a WAIT CLEAR on the semaphore. Where the number of stuck semaphores is small, their identities are definitely known, and only user semaphores are involved, this method is all that is needed.

When many semaphores are stuck, the identities of the stuck semaphores are not known, or system semaphores have become stuck, the SEMCHK utility can be used to resolve the situation.

THE SEMCHK UTILITY

SEMCHK is a utility for determining the state of any or all semaphores, and for clearing any or all semaphores.

To use SEMCHK, first access the Transient Functions Menu and select #3, "Extended Input". SEMCHK is available only when Extended Input is enabled. The facilities of Extended Input can be used when replying to any SEMCHK prompt.

Next access the Multi-User Menu and select #6, "Examine/Clear User Semaphores".

SEMCHK will become active and display the SEMCHK menu, which is:

```
1.  List all semaphores currently set.
2.  Clear all semaphores.
3.  Test a specific semaphore.
4.  Clear a specific semaphore.
5.  Exit                ? 5
```

where the the "5" indicates that the cursor remains at that position. The "5" is a default response, and is offered each time the SEMCHK menu is displayed.

A default response is printed after every SEMCHK prompt for which one is appropriate. To take the default, just hit <cr>. To enter some other response, type it in; the default response will be overwritten.

SEMCHK Device Requests

When any selection but EXIT is taken from the SEMCHK menu, SEMCHK first asks for a Device Letter. The default selection is always the letter of the device specified in the most recent DEV command. For example:

```
DEVICE ? E
```

If SEMCHK is being run on a workstation, and the semaphores of interest are local to that workstation, reply "A". Otherwise, reply with Device letter of the hard disk of the computer whose semaphores are of interest.

The results when each of the SEMCHK menu selection is described below.

List All Semaphores Currently Set

This selection produces a dynamically updated display giving the state of every semaphore.

First enter a Device Letter; see above under "SEMCHK Device Requests".

SEMCHK then asks for the lowest and highest semaphore numbers to be listed. It first prints:

```
Starting semaphore number? 001
```

Enter the desired number, or <cr>. SEMCHK then prints:

```
Ending semaphore number? 230
```

Again enter the desired number or a <cr>.

Once it has the information it needs, SEMCHK clears the screen and begins scanning the semaphores of interest while sweeping the cursor across the screen. When SEMCHK encounters a set semaphore, it prints the number of that semaphore on the screen. The number of a given semaphore is always displayed at the same screen location when that semaphore is set. When SEMCHK encounters a cleared semaphore, it prints three blanks, which over-writes the number of any semaphore that has been cleared since the last scan.

The resulting display is remeniscent of a radar screen, with semaphore numbers blinking on and off accordingly as the actual semaphores are set and cleared. The usefulness of this display is not restricted to the identification of stuck semaphores; it can be used for any purpose that requires real-time knowledge of how a computer's semaphores are being used.

The semaphore scan continues until it is explicitly cancelled. To cancel it, enter control-C. The SEMCHK menu will be repeated.

Clear All Semaphores

First enter a Device Letter. See above under SEMCHK device requests.

Once it has a Device Letter, SEMCHK begins clearing user semaphores on the computer containing the device. It begins at Semaphore 0, and clears semaphores at the rate of about 3 per second until it reaches Semaphore 255.

The clearing process can be stopped at any time by hitting control-C. When control-C is hit or Semaphore 255 has been cleared, SEMCHK repeats its menu.

Test a Specific Semaphore

SEMCHK first asks for a Device Letter. See above under "SEMCHK Device Requests".

Next SEMCHK asks for the number of the semaphore to be tested.

```
Semaphore number to test?
```

No default number appears. Enter the number of the semaphore of interest.

When it has a semaphore number, SEMCHK begins testing the semaphore repeatedly. Each time it finds the semaphore set, it prints:

```
Semaphore n is set:
```

where n is the number of the semaphore. Each time SEMCHK finds the semaphore clear, it prints:

```
Semaphore n is not set
```

The cycle of testing and displaying the result continues indefinitely. To terminate it, enter control-C. The SEMCHK menu will be repeated.

Clear a Specific Semaphore

First respond to SEMCHK's request for a Device letter, as described above under "SEMCHK Device Requests".

SEMCHK next prints:

```
            Semaphore number to clear?
```

with no default selection. Enter the number of the semaphore to be cleared.

SEMCHK then clears the semaphore. No error results if it was clear already, and no verification message appears. When the semaphore is clear, SEMCHK repeats its menu.

IMPLEMENTING THE HARD DISK

Procedures for partially implementing the hard disk are described in Chapter Two of this manual. Topics covered there are the initialization of one megabyte of the disk, configuration of the floppy diskettes with the SHSET and DSKSET programs (CD-28, 36 and 74 systems), and installation of the hard disk software. The partial description was necessary to accomplish two purposes; to provide a second device on which to learn OS-65U, and to provide a preliminary test of the disk's first megabyte of space

This chapter describes the procedures required to bring the hard disk to full implementation. Procedures are covered for testing the disk, initializing it fully, identifying/isolating any defective sectors that may be on it, and installing all OS-65U software, including the multi-user files.

If your system uses two hard disks, it is best to take each of them individually through all the procedures described in this chapter.

NOTE

If your floppy based diskette has not been configured with the SHSET or DSKSET programs, it is necessary to do so before executing any of the procedures described in this chapter. Instructions are given in Chapter Two.

TESTING THE DISK (SHTSO7, SHTS23, OKTEST)

The first step in implementing the hard disk is to test it for defective sectors. Defective sector information is used as a basis for analyzing the disk, if necessary, and for entry into a defective sectors file (DEFILE) which is described later in this chapter.

PROMPT	RESPONSE
From Address (<cr>=0)?	<cr>
To Address (<cr>=7340031)?	<cr>
Will test through 7340031. Alright (Y/N)?	Y
Check your watch. This will take 9.6 minutes	

An explanation of the prompts follows:

Omit re-tries? If a bad sector is encountered, should the program re-try it (10-12 times) before reporting an error, or should bad sectors be reported when first encountered? This option can slow down the test execution, but provides greater confirmation that the sector is actually bad. The recommended procedure is to omit the re-tries, if errors are reported, test the disk a second time using re-tries. If errors reported during the first test do not appear on the second test, disregard them.

Pause after an error? If an error is encountered, should program execution stop until told to continue (by pressing the <Return> key)? The purpose of this option is to allow enough time to record the data about each error. It is very important that all data be recorded about each error. This option should generally be used. It is unnecessary, however, if using the following option.

Print error messages on printer and console? This option prints all error data on the system printer while simultaneously displaying it on the console. This option generally eliminates the need for the previous option (above). Be certain the printer is on-line and operational.

After the prompts and responses, the program tests the disk with a Read/Write operation on each of its sectors. Each cylinder is displayed on the screen as it is verified. Error messages are displayed as defective sectors are encountered; data about them must be recorded (manually or on the printer). Error messages display the following information:

DEVice error occurred on.
Type of error, e.g., checksum, overflow, parity, etc.
Address of error
Cylinder address of error.
Track (Head) error occurred on.
Sector number of error.
Subroutine error occurred in.

At this point, it is not necessary to understand or analyze the error data, just record it. As an example, the following defective sector message could occur:

```
DEVice E -- HEADER CHECKSUM -- ERROR AT ADDRESS 2021376
(Cylinder=123, Track(Head)=2,Sector=3)IN THE WRITE SUBROUTINE
```

To begin the testing procedure, the system must be booted. If the hard disk software has already been copied onto the hard disk via procedures in Chapter Two, (or has been installed by the factory or dealer), boot from the hard disk. If the hard disk does not contain an OS-65U operating system, boot from the Floppy-Based diskette. If necessary, see the System Startup instructions in Chapter Two.

CAUTION

Testing the disk is destructive to information on it -- everything on the disk will be erased. Check with your dealer before proceeding. The disk may have already been tested, and may contain proprietary software.

You will find the test programs SHTS07, SHTS23 and OKTEST are on the Time Share Files diskette. After booting the system, insert the Time Share diskette in DEVice A. Masterkey model numbers are listed below with the proper test program for each. As a double check, the test programs verify that the right test is being run for the given disk and return an error message if it is incorrect.

Model		Test Program
230E	(CD-7)	SHTS07
230I	(CD-28)	SHTS07
250I	(CD-36)	OKTEST
250J	(CD-74)	OKTEST
Challenger	(CD-23)	SHTS23

To run the test program, select a program from the table (above) and RUN it as described below; a sample conversation is given for testing a 7-megabyte system. Note that most of the prompts have a default (<cr>) response that can be invoked with the <Return> key. The prompts are explained after the conversation. From immediate mode, enter

```
RUN"SHTS07","PASS"
```

and respond appropriately to the prompts.

PROMPT	RESPONSE
*** NOTICE:THIS IS A DESTRUCTIVE TEST***	
Continue only if you are willing to destroy the data on your hard disk!!!! Continue (Y/N)?	Y
Omit retries (<cr>=Y)?	<cr>
Pause after an error (<cr>=Y)?	<cr>
Print error messages on printer and console (<cr>=Y)?	<cr>

NOTE

If more than five (5) errors are reported during a disk test (using re-tries), the errors should be evaluated. Contact your dealer or Ohio Scientific.

When the test is finished, the program will return to immediate mode.

INITIALIZING THE DISK (COPIER)

After the hard disk has been tested according to instructions in the previous section, it is ready to be fully initialized. Before initializing, be certain that the Floppy Diskette has been configured with the SHSET or DSKSET programs (Chapter Two).

CAUTION

Initializing the disk is destructive to information on it--everything on the disk is erased. Check with your dealer before proceeding. The disk may already be initialized--and may contain proprietary software.

Insert the Floppy-Based diskette into DEVice A and boot the system from the diskette. Select the "System Utilities" menu from the Main Menu and select "Disk Copier" from the Systems Utilities menu. From the Disk Copier menu, select (I)nitialize. The prompts listed below will appear sequentially. Sample responses (for a 36-megabyte system) are given.

PROMPT	RESPONSE
DEVice?	E
Password?	3300
From Address?	0
To Address?	36449280
Will initialize 0 through 36449280	
Allright?	Y
Wait nn.n minutes	

The floppy diskette may be removed while the disk is initializing. When finished, the Disk Copier menu will reappear.

System passwords may be found in Appendix E. The "To Address" prompt must be responded to with the formatted disk size. They are as follows:

Model No.		Formatted Disk Size
230E	(CD-7)	7340032
230I	(CD-28)	29360128
250I	(CD-36)	36449280
250J	(CD-74)	72898560
Challenger	(CD-23)	23166976

INSTALLING THE OS-65U SOFTWARE

After the disk has been initialized, the OS-65U hard disk based software may be copied onto it. Before proceeding, check with the dealer to determine if proprietary software has been installed on the disk. Before installing the hard disk software, be certain the floppy based and hard disk based diskettes have been configured to the disk with the SHSET or DSKSET program (see Chapter Two).

Follow the steps below to copy the hard disk software to the hard disk:

1) Reset the computer with the RESET key on the front.

2) Insert the Floppy-Based diskette in DEVice A.

3) Boot the system from the floppy diskette (respond with a "D" to the "H/D/M" prompt).

4) Select "System Utilities" from the Main menu.

5) Select "Disk Copier" from the System Utilities menu.

6) Remove the Floppy-Based diskette and insert the Hard Disk-Based diskette.

7) From the Disk Copier menu appearing on the screen, select (B)oth. Respond to the prompts as follows:

PROMPT	RESPONSE
From DEVice?	A
To DEVice?	E
To system based at address?	0
Are you sure?	Y

The program then copies the contents of the Hard Disk-Based diskette one sector at a time and returns to the Disk Copier menu.

NOTE

No errors should occur while the diskette is copying.

If a Read error occurs, remove/insert the diskette and try again.

If an error is reported, try another copy of the diskette. If a Read error is then reported, contact your dealer or Ohio Scientific.

If a Write error occurs, re-initialize 1 megabyte of the hard disk (see instructions in Chapter Two) and try to copy the diskette again. If a Write error is then reported, contact your dealer or Ohio Scientific.

MAINTAINING THE DEFECTIVE SECTORS FILE (DEFLIS)

After the OS-65U operating system and programs have been copied onto the hard disk, it is necessary to enter the addresses of the defective sectors into the defective sectors file, DEFILE. DEFILE is updated and maintained by the program DEFLIS.

NOTE

If no errors were reported during the disk test, this section may be skipped.

DEFILE AND THE CREATE PROGRAM

The defective sectors file, DEFILE is examined by the CREATE program before a file is created in order to avoid creating files over defective sectors. If the file to be created will begin on a defective sector, CREATE creates a dummy file over the sector and creates the desired file beginning at the next sector. If a defective sector occurs elsewhere than on the file's beginning sector, CREATE makes a blank filler file up to the defective sector, a dummy file over the defective sector and starts the file at the next sector. Defective sector dummy files and filler files are listed on the disk Directory by the names !Dnnn and !Fnnn respectively. Examples of each follow:

Name	Type	Access	Address	Length	Sec Bnd	Sec Len
!D0167	Other	R/W	598528	3584	Yes	Yes
!F0170	Other	R/W	602112	10752	Yes	Yes

Filler files can be reused, either by the CREATE program with a program its same size, or by being RENAMEd by the user.

CAUTION

Do not replace a disk with defective sectors with the PACKER program.

RUNNING THE DEFLIS PROGRAM

To update the defective sectors file with the DEFLIS program, reset the machine and boot from the hard disk (respond with an "H" to the "H/D/M?" prompt). From immediate mode, enter the command:

```
RUN"DEFLIS","PASS"
```

The system then issues the following prompt:

```
List(L),Add(A),(C)lear File or Exit(X)?
```

The addresses of defective sectors encountered during the disk test (see earlier section of this chapter) must now be entered. To do that, select "(A)dd" which produces the following prompt:

```
Address of defective sector or 0 if done?
```

Enter the address of the first defective sector, and press the <Return> key. The prompt will reappear after each entry until "0" is entered.

The addresses being entered are placed directly into DEFILE. Should DEFILE need to be cleared for any reason (incorrect address entry, etc.), press "0" in response to the last prompt (above). The previous prompt (above) will reappear. Select "(C)lear File" to clear all address entries from DEFILE.. You may now begin entering addresses again. The "(L)ist" option may be used at any time to list the addresses that have been entered.

When all the defective sector addresses have been entered, DEFILE has been updated. The "(X)" selection exits the program and returns you to the immediate mode.

INSTALLATION OF OS-65U TIMESHARING SOFTWARE

Before you read this chapter, it is essential that you read Chapter 10, "Keyware OS-65U Timesharing", in its entirety. Many terms and concepts used in this chapter are defined in Chapter 10, and the definitions are not repeated here.

This chapter tells you how to install and configure the software required for OS-65U Timesharing.

This chapter does not tell:

- How to install and configure the hardware needed for timesharing. See your OSI dealer for information about hardware installation
- How to start and use OS-65U Timesharing. That information is given in Chapter 10.

TIMESHARING HARDWARE ENVIRONMENT

OS-65U Timesharing requires the following hardware:

1. A Masterkey 230 or 250 series computer.
2. A CM-20 48K Timesharing Memory Partition Board for each timesharing user (already exists for User 0).
3. A terminal and RS-232 cable for each timesharing user.

For information about terminal specifications for any Masterkey computer, see the appropriate Hardware User's Guide.

More on the CM-20 Board

A Masterkey computer is initially equipped with 56 kilobytes of main memory, which is sufficient to support one user. For each additional user, an additional 48K of memory is needed. This memory is supplied by installing one CM-20 Timesharing Memory Partition Board for each additional user. A 230 can hold up to 3 such boards; a 250 can hold up to 5.

A set of switches on each board sets the partition/user number for that board. This number corresponds to the timesharing user number that appears next to the terminal connector for that user on the rear panel.

TIMESHARING SOFTWARE

Timesharing on a Masterkey computer requires the following software:

1. OS-65U Version 1.43 or later.
2. The files MMENU* and LEVEL3.
3. The file TSCD07 for a 230 series computer, or TSCD36 for a 250 series computer.

OS-65U is on the OS-65U System Diskette. MMENU*, LEVEL3, TSCD07, and TSCD36 are on the OS-65U Timesharing Diskette. Be sure the version numbers on the diskettes match, or the system will not work correctly.

INSTALLING THE TIMESHARING SOFTWARE

The following are steps needed to install the OS-65U Timesharing software. These steps assume that the computer's hard disk has been initialized, and that the standard OS-65U system software has been installed on it. These operations are described in Chapter 15.

1. Boot the computer from the hard disk.
2. Use the INSTAL program as described in Chapter 15 to copy the Timesharing Diskette to Device E File System 0. INSTAL will leave you in Immediate Mode when it has finished executing.
3. A timesharing executive program must be modified to reflect the number of timesharing users there will be. The program to be modified depends on series of the computer on which timesharing is being installed, as follows:

Series	Program
230	TSCD07
250	TSCD36

   Load the appropriate program by typing:

```
LOAD "name-of-program"
```

4. Modify the program to reflect the number of timesharing users, by typing:

```
352 TS=number-of-users:POKE 54352,TS*2
```

   where number-of-users is the number of timesharing users the system will have, including User 0.

5. Save the modified program by typing:

```
RUN 63999
```

The program will be saved as directed by the code at Line 63999.

Installation of the software necessary for timesharing is now complete. Once both hardware and software have been installed, timesharing can begin.

MODIFYING A TIMESHARING SITE

When the number of timesharing users changes, the timesharing software must be modified to indicate the new number. To make this change, enter Immediate Mode and redo steps 3-5 above under "Installing the Timesharing Software".

INSTALLATION OF OS-65U PRINT SPOOLING SOFTWARE

Before you read this chapter, it is essential that you read Chapter 11, "Keyware OS-65U Print Spooling", in its entirety. Many terms and concepts used in this chapter are defined in Chapter 11, and the definitions are not repeated here.

This chapter discusses:

- Software required for Print Spooling
- Installing the Print Spooling Software

No special hardware is required for Print Spooling.

This chapter does not tell you how to start and use Print Spooling. That information is in Chapter 11.

PRINT SPOOLING SOFTWARE

Print spooling requires the following software:

1. OS-65U Version 1.30 or later
2. One Device E, the files MMENU*, SPOOL1, SPOOL2, DSPOOL, SPOOLC, SPL 0, SPL1, SPL2 and SPL3.

There may also be up to 12 additional spool files, SPL4-SPL15, on Device E.

If Device E contains more than one file system, all the files listed must be present on every file system on which Print Spooling will be done.

INSTALLING THE PRINT SPOOLING SOFTWARE

Installing OS-65U Print Spooling requires four basic steps:

1. Install the programs that perform Print Spooling.

2. Create the spool files SPL1-SPL3, and optionally SPL4 up to SPL15.

3. If more than three spool files were created in Step 2, modify the SPL 0 file to reflect the number of spool files created.

4. Enable Automatic Despooler Startup if desired.

Steps 1-3 must be performed on every OS-65U File System from which Print Spooling will be done. In the following descriptions, the file system on which Print Spooling is being installed is referred to as the current file system. The descriptions assume that the computer's hard disk has been initialized, and that the standard OS-65U system software has been installed on it, as described in Chapter 15.

Install the Print Spooler Programs

If a Device E has been configured for timesharing, this step can be omitted for File System 0 of that device. The Print Spooler programs are supplied on the Timesharing Diskette, and so were installed automatically when the Timesharing Diskette was copied to File System 0.

Otherwise, use CREATE and COPYFI to copy the following files from the Timesharing Diskette to the current file system. The size of each file is shown.

File	Size (Bytes)
SPOOL1	10752
SPOOL2	10752
DSPOOL	10752
SPOOLC	10752
SPL 0	7168

Create the Spool Files

If a Device E has been configured for timesharing, there are already three dummy spool files named SPL1, SPL2 and SPL3 on File System 0. Before you create any spool files on such a file system, these dummy files must be deleted. The method for deleting them is described in Chapter 2. They are R/W files, so the password needed for deletion is ".".

There must be at least three and may be up to 15 spool files on a file system. Create the desired number of files with the CREATE utility (Chapter 2). Each file must be named SPLn, be of type DATA, and have access rights of R/W. The numbers used in the file names must be a consecutive series starting with 1. Stated differently, SPL3 must exist, and if SPLn exists, SPLn-1 must exist also.

There is no simple formula for determining the optimal size for a spool file, because applications for spooling differ so widely. The following facts may be helpful in estimating the size:

1. **When data is spooled, repeated characters are compressed into a two byte field, consisting of one example of the character followed by a count of the number of times the character is repeated. Spooled data is decompressed before it is printed.**

2. Every line of data is followed by two invisible bytes, a carriage return and a line feed.

In general, spool files should be at least 50,000 bytes long unless there can be no question that a smaller size will be sufficient. There is no upper limit to the size of a spool file: The only requirement, as with any file, is that sufficient free disk space be available when the file is created.

All the spool files on a given file system should be the same size. There is no point in making them different sizes, because there is no way to choose a spool file of the right size for a particular spool job.

Modify the SPL 0 File (if necessary)

If there are only three spool files on the current file system, this step may be skipped. The SPL 0 file as shipped is configured for a file system with three spool files.

If there are more than three spool files, each file after the third requires modification of the SPL 0 file to make the spool file available for spooling and despooling. The modification consists of adding information about the spool file to the SPL 0 file.

The SPL 0 file is modified by using the Spool Job Manager's Operator Interface, which is described in Chapter 11 under "Controlling the Spool Job Manager". Be sure you are familiar with that section before proceeding.

First enable Extended Input. The Manager's Operator Interface is available only when Extended Input is enabled. Then access the Multi-User Menu and select #9, "Examine/ Modify Spool Control File".

The system will then print the Spool Job Manager Display (see Chapter 11). Perform the following steps for each spool file after the third.

1. From the Spool Job Manager Menu select #3, "Examine Job".

2. The system will respond

   Enter password:

   Type the required password, which is "PASS".

3. The system will print

   Enter number of job to examine:

   Enter the number following the "SPL" in the name of the spool file currently being added to SPL 0, i.e. 4 for SPL4, 5 for SPL5, etc.

4. The system will print the Spool Job Parameter Display (see Chapter 10) followed by the prompt

   Enter field to be changed:

   This display and prompt will be repeated after each of the following changes is made. The capabilities of Extended Input are available for making these changes.

5. Change the file name from "Blank" to its actual name, i.e. "SPL4", "SPL5", etc.

6. Change the status from "Blank" to "E".

7. Leave the priority unchanged; its value is always "9".

8. Change the job name from "Blank" to a null string, by deleting the "Blank" with the DELETE key, then typing one blank followed by <cr>.

9. Leave the output device unchanged; its value is always "5".

10. Enter a <cr>. The Spool Job Manager Display will be repeated, and the next spool file (if any) can be recorded in SPL 0 by starting over at Step 1.

11. When the last spool file has been entered, select #4, "Exit", from the Spool Job Manager Menu. The Multi-User Menu will be displayed.

When the preceeding steps have been carried out, installation of the software necessary for Print Spooling is complete, and Print Spooling can begin. An additional step is required if Automatic Despooler Startup is to be used.

ENABLING AUTOMATIC DESPOOLER STARTUP

Automatic Despooler Startup can run only on a timesharing partition other than Partition 0. It can run from any Device E file system .

If Automatic Despooler Startup will be used, the copy of BEXEC* on the file system from which the despooler will print must be modified. The modification requires only one change to BEXEC*. Enter Immediate Mode, then type

```
LOAD    "BEXEC*"
1543    SP=timehsaring-partition-number
RUN     63999
```

where timesharing-partition-number is the number of the partition on which the Despooler will run.

MODIFYING THE PRINT SPOOLING UTILITY

To add a spool file after Print Spooling has been installed, create the spool file and modify the SPL 0 file as described above under "Create the Spool Files" and "Modify the SPL 0 File". There is no special significance to the fact that the file was not created at installation time.

To change the size of a spool file, delete it, then replace it with a file having the same name and the desired size. The SPL 0 file need not be modified.

To delete a spool file, first delete the file itself. Then go through the "Modify the SPL 0 File" procedure given above, and set the file's name, status, and job name to "Blank".

Automatic Despooler Startup can be enabled or disabled at any time. To enable it, follow the instructions above under "Enabling Automatic Despooler Startup". To disable it, select the file system that contains the copy of BEXEC* that was modified to enable it, then type

```
LOAD    "BEXEC*"
1543    SP=16
RUN     63999
```

INSTALLATION OF OS-65U NETWORKING SOFTWARE

Before you read this chapter, it is essential that you read Chapter 12, "Keyware OS-65U Networking", in its entirety. Many terms and concepts used in this chapter are defined in Chapter 12 and the definitions are not repeated here.

This chapter tells you how to install and configure the software required for OS-65U Networking.

This chapter does not tell:

- How to install and configure the hardware needed for networking. See your OSI dealer for information about hardware installation.
- How to start and use OS-65U Networking. That is discussed in Chapter 12.

This chapter assumes a network with two controller nodes, each of which has both workstations and timesharing users. However, the instructions given can be used for any network. Any step that applies only to network components not used at a particular site can be skipped when networking is installed at that site.

NETWORK HARDWARE ENVIRONMENT

A network requires the following hardware:

1. One or two Masterkey 230 or 250 Series computers to serve as Network Controller Nodes. The controller nodes need not be of the same type or series.
2. A UK-WS-230 Network Workstation Support kit for each controller node that is a Masterkey 230.
3. A UK-WS-250 Network Workstation Support kit for each controller node that is a Masterkey 250.

4. For each controller node, up to 8 complete 230, or 250 series computers, to serve as workstations. The workstations need not be of the same type or series.

5. For each controller node and workstation, a terminal and an RS-232 cable.

6. For each controller node that will also run timesharing users other than User 0, all timesharing hardware listed in the previous chapter for a computer of its type.

7. If there are two controller nodes a Keyring cable to connect them.

8. For each workstation, a cable to connect it to its controller node. For information about physical characteristics, terminal specifications, and hookup requirements for any Masterkey computer, see the appropriate Hardware User Guide.

More on the Workstation Support Kit

A Workstation Support Kit converts an OSI 230- or 250-series computer into a controller node, by providing the additional hardware capabilities the computer needs to support workstations. A Workstation Support Kit contains two major components:

- A Workstation Connector Box (for a 230) or Panel (for a 250), which provides the connectors necessary to plug cables from workstations into the controller node.

- A Network Control Board, which provides the circuitry that actually ties the separate computers together into a network.

For information on how to install a Workstation Support Kit, see the instructions that come with the kit.

KEYRING SOFTWARE

A Keyring network requires the following software:

1. OS-65U Version 1.43 or later

2. The files MMENU* and LEVEL3

3. The file TSCD07 for a 230 series computer or TSCD36 for a 250 series computer.

4. The files NETWRK, TSNET, L2NET, C2NET and CONSOL.

OS-65U is on the OS-65U System Diskette. MMENU*, LEVEL3, TSCD07 and TSCD36 are on the OS-65U Timesharing Diskette. NETWRK, TSNET, L2NET, C2NET and CONSOL are on the OS-65U Network Diskette. Be sure the version numbers on all three diskettes match, or the system will not work correctly.

INSTALLING THE KEYRING SOFTWARE

The following are the steps needed to install the Keyring network software. These steps assume that the computer's hard disk has been initialized and that the standard OS-65U system software has been installed on it, as described in Chapter 15.

Installing the software required for networking includes installation of all software required for timesharing. If either controller node is already configured for timesharing, steps marked "*" in the following should be omitted for that station.

Installing Networking on a Controller Node

1. Boot the controller node from the hard disk.

*2. Use the INSTAL program as described in Chapter 9 to copy the Timesharing Diskette to Device E, File System 0.

3. Use the INSTAL program to copy the Network Diskette to Device E, File System 0.

4. A timesharing executive program must be modified to indicate

   - *The number of timesharing users.
   - The number of workstations.
   - The type of computer that the controller node is.

   The program to modify depends on the series of the computer on which networking is being installed, as follows:

Series	Program
230	TSCD07
250	TSCD36

   Load the appropriate program by typing:

   ```
   LOAD "name-of-program"
   ```

*5. Indicate the number of timesharing users, by typing:

   ```
   352 TS = number-of-users: POKE 54352, TS*2
   ```

   where number-of-users is the number of timesharing users the control station will have, including User 0.

6. Similarly indicate the number of workstations, by typing:

   ```
   280 PTS = number-of-workstations
   ```

7. Save the modified program, by typing:

```
RUN 63999
```

   The program will be saved as directed by the code at Line 63999.

8. The program NETWRK must be modified to indicate:

   - The Network Device Letter that the controller node will have during networking.
   - The Network Device Letter that the other controller node (if any) will have during networking.
   - The types of message to be printed at the Network Console.
   - The speed at which the network will run.

   To begin the modification, remain in immediate mode and type:

```
LOAD "NETWRK"
```

9. Indicate the Network Device Letter by which the controllernode's Device E (its hard disk) will be known during networking. The letter must be K or L. Type:

```
1110 ID$ = "device-letter"
```

10. Indicate the Network Device Letter by which the other controller node's Device E will be known. If K was given in Step 9, L must be given now, and vice versa. Type:

```
1132 FAR$ = "device-letter"
```

   If there is no other controller node, give a null string for its device letter:

```
1132 FAR$ = ""
```

11. Indicate the types of message you want to appear on the Network Console.

   To turn semaphore messages on, type:

```
1211 POKE 25769,0
```

   To turn them off, type:

```
1211 POKE 25769,6
```

   To turn transaction messages on, type:

```
1210 DV=1:POKE 24645,0
```

To turn them off, type:

```
1210 DV=1:POKE 24645,189
```

Error messages are always on; they cannot be turned off.

12. Indicate the speed at which the network will run. All computers in the network must run at the same speed. If the network will run at 2MHz, type:

    ```
    977 SW=-1
    ```

    If the network will run at 1MHz, type:

    ```
    977 SW=0
    ```

13. Save the modified NETWRK routine by typing:

    ```
    RUN 63999
    ```

14. If Automatic Network Startup will be used, the routine BEXEC* must be modified to indicate this fact and to designate the timesharing partition on which the Network Manager is to run.

    To accomplish these modifications, only one change to BEXEC* is required. Type:

    ```
    LOAD "BEXEC*"
    1530 NS = timesharing-partition-number
    RUN  63999
    ```

    where timesharing partition number is the number of the partition on which the Network Manager will run. This number must not be 0.

15. The routine L2NET must be modified to indicate the speed at which the network will run. First type:

    ```
    LOAD "L2NET"
    ```

    If the network will run at 2 MHz, type:

    ```
    152 SW=-1
    ```

    If the network will run at 1 MHz, type:

    ```
    152 SW=0
    ```

    Then save the modified routine by typing:

    ```
    RUN 63999
    ```

Installing Networking on a Workstation

The final stage of network software installation is to copy the modified L2NET routine created in Step 15 onto every OS-65U system diskette and device E file system that will be used by a workstation to access the network. The size of the L2NET program file is 21504 bytes. L2NET is the only program a workstation needs to accomplish network access.

Installation Complete

When the preceding steps have been carried out, installation of the software necessary for networking is complete. Once both hardware and software have been installed, networking can begin.

Modifying a Network

When the hardware of a network is modified in any way, the software must be modified accordingly. To modify the software, go through the "Installing the Network Software" sequence again, but perform only those steps that specifically concern the hardware changes that have been made.

Automatic Network Startup can be enabled or disabled at any time. To enable it, enter Immediate Mode, then perform Step 14 above under "Installing the Network Software". To disable it, enter Immediate Mode, then type:

```
LOAD     "BEXEC*"
1530     NS=16
RUN      63999
```

APPENDICES

OS-65U SUMMARY OF PROGRAMS

The Keyware OS-65U operating system programs and files are listed alphabetically below, with function description, execution method(s) and the chapter in the manual where a full description of the program may be found. Execution methods are: Main Menu, System Utilities Menu (S/U), Transient Functions Menu (T/F) or Immediate mode (Immed). Syntax for Immediate Mode execution is RUN"nnnnnn" where nnnnnn is the file name.

File Name	Function	Execution	Chapter
BEXEC*	Initializes system, displays Main menu	Immed	2
CHANGE	Modify disk contents	Immed	9
COMKIL	Common Variables	T/F Menu	6
COPIER	Copies disk(ette) -- all or portion	S/U Menu, Immed	2
COPYFI	Copies file	S/U Menu, Immed	2
CREATE	Creates storage files	S/U Menu, Immed	2
CRT 0	Stores CRT codes	S/U Menu, (DMS+)	7
CRTSET	Terminal Setup	T/F Menu	6
DEFILE	Stores defective sector address information	N/A	9
DEFLIS	Maintains DEFILE	Immed	9
DELETE	Deletes file	S/U Menu, Immed	2
DIR	Examines DIREC* file	Main, S/U Menus, Immed	2
DIREC*	Disk directory	N/A	2
ED	Keybase File Editor	S/U Menu, Immed	7
EDITOR	Program line editor	T/F Menu	6
FDUMP	Dumps file contents	S/U Menu, Immed	7
FPRINT	Print data file contents	Immed	7

GETCRT	Retrieves CRT parameters	Immed	7
INP$	Enables Extended Input	T/F Menu	6
INPOUT	Disables INP$	T/F Menu, Immed	6
INSTAL	Installs system files to disk	Immed	9
MMENU*	Displays Multi-User Menu	Main Menu, Immed	9
PACKER	Reclaims deleted disk space for re-use.	Immed	8
PRTMAP	Configures system printer	Immed	8
PRTSET	Configures devices 3 & 8	Immed	App. F
RENAME	Changes file name, password, access, etc.	S/U Menu, Immed	2
RSEQ	Program line resequencer	T/F Menu	6
SYSDIR	Defines multiple systems	Immed	9
WPDRIV	Parallel WP Driver	Immed	8
//	System Utilities Menu	Immed	2
/	Transient Functions Menu	Immed	2

MENU TREE BY PROGRAM NAME Figure A-1 illustrates the programs as structured in the OS-65U menus.

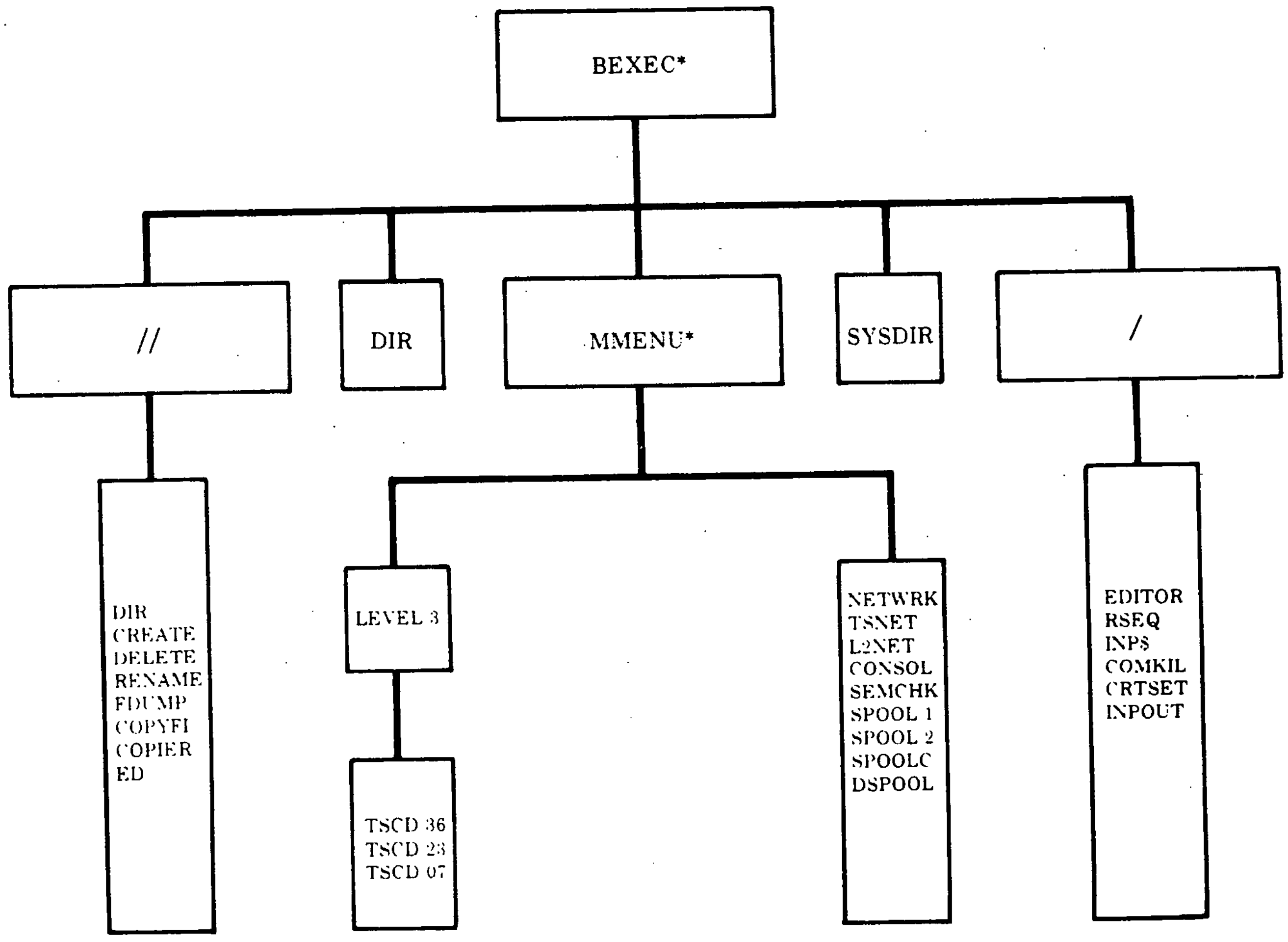

FIGURE A-1

BASIC COMMANDS AND RESERVED WORDS

Command	Syntax	Description
ABS	ABS (X)	Returns the absolute value of expression X.
AND	X AND Y	Logical AND operator.
ASC	ASC(X$)	Returns the decimal ASCII value of the first character in the string X$.
ATN	ATN(X)	Returns trigonometric arctangent (X). Result is in radians in range -1 to 1. Not available when INP$ transient utility is enabled.
CHR$	CHR$(I)	Returns a one character string, whose decimal ASCII value is that (I).
CLEAR	CLEAR	Clears the variable table and RESTOREs the DATA pointer.
CLOSE	CLOSE CLOSE n	Closes an open disk file by dumping the disk buffer and with no channel number specified closes all open channels. A CLOSE n command, where n is a channel that has not previously been opened, will produce an error.
CONT	CONT	Continue execution of a program that has been halted either by a control C key-in or a STOP statement.
COS	COS(X)	Trigonometric cosine function with the argument in radians. Not available when INP$ transient function is enabled.
DATA	DATA 4,72,"HI"	Provides data elements for READ statements. Strings may appear quoted or unquoted. If unquoted, leading blanks are ignored and trailing blanks are included.
DEF	DEF FNA(X)= X+SIN(X)	Define function statement,where A and X are simple variable names. Not available when COMKIL transient utility is enabled.

DEV	DEV 'A' DEV D$	Specifies which disk DEVice is to be currently on-line. The argument must be a single character string literal or variable. The possible values for DEV are: Single User: A-D,E Intelligent Terminal: A-D,E, K-Z Time Sharing: A-D,E. Network: A-D,E,K-Z.
DIM	DIM A(20), B$(10,5)	Dimension statement to allocate space for subscripted variables.
END	END	Terminate program execution. This statement need not appear in a program at all, nor be the last statement in a program.
EXP	EXP(X)	Exponential function of e (2.71828...) raised to the power of (X). Not available when INP$ transient utility is on-line.
FIND	FIND "LOAN",2 FIND A$, CH	High speed search for the string expression (first argument) in the disk file opened on channel number (second argument). Search starts at current INDEX of that channel. If found, returns INDEX of the found location in the file, otherwise returns INDEX value of greater than or equal to 1E9.
FLAG	FLAG 3	Enables the system option defined by the flag number.
FN	DEF FNA(X)=2+X	Function name, of the form FN followed by a variable name. Not available when COMKIL transient utility is enabled.
FOR	FOR I = 1 TO 5 FOR J = a TO B STEP C	FOR-NEXT loop range definition verb. Allows repeated execution of same statements.
FRE	FRE(X)	Returns the number of bytes of memory workspace available that are unused. X is a dummy variable.
GOSUB	GOSUB 150 ON X GOSUB 10,70 GOSUB A	Execute a BASIC subroutine beginning at the line number equal to the numeric argument.

GOTO	GOTO 150 ON X GOTO 100,200	Unconditional transfer of program execution to the line number equal to the argument.

IF	IF X = 1 THEN GOTO 50
	IF I>1 THEN I=1
		Conditional statement execution verb.

INDEX	INDEX<CH>=0 N=INDEX(CH)	Set or equate an open disk channel's file index position. INDEX<n> is an index assignment, INDEX(n) is an index equate. In an INDEX<n>=x, the value x must be a non-negative integer.
INPUT	INPUT A	Cause BASIC to request data from console keyboard, or from specified input device or disk file. The INPUT [ ] form of this command is available only with INP$ transient utility enabled.
INPUT	'NAME';N$ INPUT#1, B INPUT%3, F$ INPUT [3, "A"]QA$	

INT	I = INT(X)
	I = INT(3.1415)
		Return the greatest integer less than or equal to the numeric argument. Note that INT(-2.1) equals -3.

KILL	KILL A, A$ KILL C( ), B$( ) KILL * KILL (*)	Eliminate variables from the program variable table.Arguments may be specific simple variables such as A, specific array variables such as C( ), * to KILL all simple variables, or (*) to KILL all array variables. Available only when COMKIL transient utility is enabled.
LEFT$	B$ = LEFT$(A$,5)	Return the leftmost substring of a given string. First argument is a string expression, second argument is a positive arithmetic expression indicating the number of characters to return.
LEN	L = LEN(A$)	Return the length of the string expression argument.
LET	LET X = Y+1	Assign the value of the expression (Y+1) to the variable (X). The word LET is optional.

LIST	LIST LIST 1-50 LIST -50 LIST 100- LIST#DV,10-20 LIST%CH	Program listing verb. Examples illustrate syntax for complete listing, listing of line number ranges, and listing to a specific output device or disk channel.
LOAD	LOAD "PROG1" LOAD "P1",'PASS"	Command to load a program from disk to memory. Arguments are program name, and password if required.
LOG	LOG(X)	Returns the natural logarithm (log to the base e) of the numeric argument. Not available with INP$ transient utility enabled.

MID$	A$ = MID$(B$,2,4)
	A$ = MID$(B$,7)
		Return a middle portionof a string argument, with a specified starting character, and a specified length -- unless to the end of the string.

NEW	NEW NEW 3584	Reset all program workspace pointers, i.e., start with a clean workspace for entry of a new program. The form NEW n, where n is a positive number, reserves an area of n bytes at the beginning of program workspace for custom programming use, such as machine language subroutines or disk transfer buffer space.
NEXT	NEXT NEXT I NEXT I,J	Terminating statement range verb for FOR-NEXT interactive loops. Jumping in and out of FOR-NEXT loop statement ranges should be avoided.
NOT	NOT X NOT (A AND B)	Logical negation operator.
NULL	NULL 8	Inserts 0 to 255 zeros (null characters) at the beginning of each string output by a LIST or PRINT command. Not available when COMKIL OR RSEQ transient utility is enabled.
ON	ON X GOSB 50, 100 ON E GOTO L1,L2,L3	Conditional transfer statement verb. In the second example, control is transferred to program line L1 if value of E is 1, and line L2 if value of E is 2, and so on.

OPEN	OPEN "FNAME", "PASS", 3 OPEN "NAE",1 OPEN FN$,PW$,CH	Open a data file on a given channel for program disk access. Arguments are file name, password (if required) and channel number (1-8).
OR	A OR B	Logical OR operator.
PEEK	C = PEEK(23468)	Returns the contents of a memory location. Argument is the memory address in decimal.
POKE	POKE 2048,199	Stores a value into a memory location. The first argument is the memory address, and the second location is the value to be stored, between 0 and 255 inclusive.
POS	POS(X)	Returns the print location of the last character printed before the call to POS. X is a dummy variable. Returns a value between 0 and 255 inclusive.
PRINT	PRINT PRINT A PRINT A,C;B$ PRINT#5"PRINTER PORT" PRINT%CH,"DISC OUTPUT" PRINT 0,"R" A$	Output commands for screen, printer, and other output device, or disk file channel.
READ	READ R< A$	Read, from DATA statements, the value of the variables appearing as arguments.
REM	REM HERE IS A COMMENT	Remark or comment initiator. All text after a REM is ignored on a given program line, i.e., it is not executed as BASIC code.
RESTORE	RESTORE	Reset the pointer in a DATA list to the first DATA item.
RETURN	RETURN	Exit verb from a BASIC GOSUB subroutine. Control is transferred to the program location immediately following the GOSUB command that initiated the subroutine execution.
RIGHT$	A$ = RIGHT$(B$,3)	Return the rightmost substring of a string argument.

RND	I = RND(X)	Return a random number between 0 and 1. If the argument is negative, it will be interpreted as a seed value. If it is zero, the function will return the last generated random number again. If the argument is positive, a random number based on the previously defined seed will be returned. Not available when INP$transient utility is enabled.
RSEQ	RSEQ NL,OL,IN RSEQ NL,OL RSEQ ,OL,IN RSEQ ,,IN RSEQ ,OL RSEQ NL,,IN RSEQ NL RSEQ	Renumbers a BASIC program Syntax reads as follows: starting at old line number OL, resequence with new line number NL, in increments of IN. Any permutation of parameters being present is permitted, nothing to keep commas as delimiters where needed. Only available when the RSEQ transient utility is enabled.
RUN	RUN RUN 200 RUN "PGM" RUN "PgM",200 RUN "PGM","PASS" RUN P$,W$,LN URN [PN$,PW$,LN]	Initiate execution of a BASIC program, either resident in memory or loaded from disk. The RUN [ ] example is the syntax for RUNning a program with saved variable values. It must have program name, password, and line number. The RUN[ ] command is only available when the COMKIL transient utility is on-line.
SAVE	SAVE SAVE "PGM" SAVE "NAME, "PASS"	Store the current program in memory onto a disk file whose name and password are the command arguments. If no arguments are given, then the disk file SAVEd to is the same as the file name used in the last LOAD command executed.
SGN	S = SGN(X=1)	Return the sign of a numeric argument, i.e., +1 for a positive valued argument, -1 for a negative valued argument, 0 for a zero valued argument.
SIN	S=SGN(AN)	Trigonometric sine function with the argument expressed in units of radians. Not available when INP$ transient utility is on-line.

SPC( )	SPC(3)	Used to print spaces inserted in output. PRINT SPC(n) will print n spaces.
SQR	R=SQR(W)	Square root function. Not available when INP$ transient utility is on-line.
STEP	FOR I = 1 TO 5 STEP @ FOR I = I TO 2 STEP -1	FOR-NEXT incremental value definition verb. Care should be taken not to explicitly assign a value to the FOR-NEXT increment variable (I is the given examples), as this may destroy the expected implicit looping.
STOP	STOP	Interrupt program execution and jump to immediate mode. The Program execution may be continued by a CONT command issued from the immediate mode.
STR$	A$ = STR$ (N)	Converts a numeric argument to its string equivalent. For example, STR$(1.2) = " 1.2", the first character being the sign (blank for positive, - for negative).
TAB	TAB(7)	Space to specified tabular position. Leftmost space is zero.
TAN	T = TAN(A)	Return trigonometric tangent function (A). A is in radians. Not available when INP transient utility is on-line.

THEN	IF A=B THEN 200
	IF C<2 THEN C=2
		Conditional statement execution directive verb.

TO	FROM I = 1 TO 3	FOR-NEXT range definition verb.
USR	Y = USR(X)	Call user-defined machine language subroutine resident in memory with argument X. The argument is a single parameter that can be passed to the user-routine.

VAL	N = VAL(A$)	Returns numeric value of a string A$, or zero if the argument is non-numeric. If the string argument is not all numeric, the effective argument becomes the substring to the first non-numeric character (e.g. VAL(123AB) equals 123).
WAIT	WAIT I,J WAIT I,J,K	Halts program execution, i.e., causes a program to "wait", until a particular bit or bits in memory is set. In the first example, the WAIT function reads the status of memory location I, then ANDs the result with value J until a non-zero result is obtained. The second example reads the status of memory location I, exclusive ORs that value with K, then ANDs that result with J until a non-zero result is obtained. This is most often used to read a character from an ACIA serial port.
WAIT CLEAR	WAIT CLEAR 2	Clears, or unsets, a semaphore in a time-sharing environment. Does not operate in a single user environment.
WAIT FOR	WAIT FOR 21	Sets a semaphore in time-sharing. Does not operate in a single user environment.

BASIC COMMANDS AND RESERVED WORDS - BY FUNCTION

The BASIC commands are listed below in functional categories.

Program Execution Control

The following commands control BASIC program execution flow such as branching, looping, and subroutine calls.

CONT	GOTO	RUN	USR
END	IF	STEP	WAIT
FLAG	NEXT	STOP	WAIT CLEAR
FOR	ON	THEN	WAIT FOR
GOSUB	RETURN	TO	

Disk Input/Output Commands

Please note that the disk I/O commands respect the access rights and file type assigned to a given file at CREATE time. If no password is designated, the system default PASS is used whenever a password is required. Possible access rights are:

Read/Write	without password.
Read	without password.
Write	without password.
None	none access without password.

The possible file types are:

BASIC	-	BASIC program file.
Data	-	General data storage.
Other	-	DIREC*, special function system files.

Consider the following example:

```
OPEN "DATFIL",1
PRINT%1,"HELLO"
```

Although the file DATFIL has read-without-password access rights, the above statements will produce an access error because the file was not opened with the system password. The correct OPEN statement would be OPEN "DATFIL","PASS",1.

Disk I/O commands follow:

```
CLOSE     LOAD
DEV       OPEN
FIND      PRINT
INDEX     SAVE
INPUT
```

General Input/Output Command

Included are commands for console, printer, memory, and disc I/O.

```
 DATA     PRINT
 INPUT    READ
*NULL     RESTORE
 PEEK     SPC
 POKE     TAB
 POS
```

Logic Functions

```
AND
NOT
OR
```

Mathematical Operations

Please note that the starred (*) functions are **not** available when the INP$ and COMKIL transient utilities are enabled.

ABS	*FN	*SIN
*ATN	ITN	*SQR
*COS	*LOG	*TAN
*DEF	*RND	VAL
*EXP	SGN	

Program Execution and Manipulation

Please note that all double-starred (**) functions listed hereafter are available **only** when the INP$ and COMKIL transient utilities are enabled.

FLAG	REM
FRE	*RSEQ
LIST	RUN
LOAD	SAVE
NEW	USR

String Variable Manipulation

ASC	MID$
CHR$	RIGHT$
LEFT$	STR$
LEN	

Variable Definition and Manipulation

CLEAR	PEEK
DIM	POKE
**KILL	REM
LET	

SUMMARY OF FLAG COMMANDS

The FLAG Commands are listed below in two ways: Numerically and by function. They enable or disable certain system features. The commands may be used either in the immediate mode or within a BASIC program in the form

FLAG n4

where n is the FLAG number.

NUMERIC LIST

For a more complete description of the FLAG commands, see the chapter in the manual.

1. Disables FLAG 2.
2. Enables the close-files-on-error and the close-files-on-immediate-mode feature.
5. Enables user programmable disk EOF action.
6. Disables FLAG 5.
7. Enables BASIC statement trace.
8. Disables FLAG 7.
9. Enables user programmable disk error action.
10. Disables FLAG 9.
11. Enables space suppression in numeric output to files and numeric to string conversions.
12. Disables FLAG 11.
13. Enables "INPUT%n," command file operation.
14. Disables FLAG 13.
15. Allows the comma and colon to be treated as valid in an INPUT or statement.

WARNING:

FLAG 15 usually renders READ statements unworkable.

16. Disables FLAG 15.
17. Disables carriage return/line feed upon terminating an INPUT or PRINT.

WARNING

All I/O devices are affected by this command.

18. Disables FLAG 17.

21. Disables FLAG 22. This Flag is overridden by FLAG 27.

22. Enables input escape on carriage return. This Flag is overridden by a FLAG 28.

23. Enables a jump to program line 50000 upon the occurrence of any or all error conditions, including BASIC and disk errors.

24. Disables FLAG 23. Enables automatic entry into immediate mode upon the occurrence of any BASIC or disk error conditions.

25. Disables FLAG 26.

26. Enables Control "C" termination of BASIC program execution.

27. Enables a null input (carriage return only) as valid INPUT sequence.

28. Disables FLAG 27. This Flag is overridden by FLAG 21.

29. Disables FLAG 30.

30. Enables a trap-overflow condition.

31. Disables FLAG 32.

32. Enables input translation.

CAUTION

If lower case is to be used in DATA statements remember to enclose the string in quotes or it will be translated.

33. Disables FLAG 34.

34. Enables output translation on LIST.

100 Performs a conditional top-of-form eject on print device #5.

101 Disables FLAG 100.

FUNCTIONAL CATEGORIES

The Flag commands are listed below in functional categories by number. Some Flags serve more than one purpose and appear in more than one category; they are indicated by an asterisk (*).

Debugging Flags

FLAG 7
FLAG 8
FLAG 30
FLAG 29

End-of-File Flags

FLAG 5
FLAG 6

Error-Handling Flags

*FLAG 2
*FLAG 1
FLAG 9
FLAG 10
FLAG 23
FLAG 24
FLAG 30
FLAG 29

Exit Flags

FLAG 26
FLAG 25

File-Closing Flags

*FLAG 2
*FLAG 1

File-Merging Flags

FLAG 13
FLAG 14

Input Flags

FLAG 15
FLAG 16
*FLAG 18
*FLAG 17
FLAG 22
FLAG 21
FLAG 27
FLAG 28
FLAG 32
FLAG 31

Input/Output Conversion Flags

FLAG 11
FLAG 12
*FLAG 18
*FLAG 17
FLAG 34
FLAG 33

Printer Control Flags

FLAG 100
FLAG 101

OS-65U ERROR CODES

Floppy Disk Error Numbers

1	- Drive not ready
2	- Seek error
3	- Invalid unit number
4	- Can't find track zero
5	- Can't find index hole
6	- Diskette write protected
7	- Track unsafe (can't verify write)
8	- Incomplete header
9	- Header: Framing Error (FE)
10	- Header: Overrun (OR)
11	- Header: OV,FE
12	- Header: Parity Error (PE)
13	- Header: PE,FE
14	- Header: PE,OV
15	- Header: PE,OV,FE
16	- Data Field: Incomplete
17	- Data Field: Framing error
18	- Data Field: Overrun
19	- Data Field: OV,FE
20	- Data Field: Parity error
21	- Data Field: PE,FE
22	- Data Field: PE,OV
23	- Data Field: PE,OV FE
24	- Checksum error
25	- Unit out of service
26	- Old 65-D header found
27	- Track 0 verification error
76	- Track out of range

CD-36 and CD-74 Hard Disk Error Numbers

1	- Drive not ready
2	- Seek timeout
3	- Invalid unit number
4	- Restore timeout
5	- DMA failed to terminate
6	- Write protect error
7	- Sector unsafe (can't verify write)
8	- Sector header checksum error
9	- Cylinder mismatch
10	- Track mismatch
11	- Sector mismatch
16	- Data field checksum error
24	- Status error
25	- Unit out of service
82	- Cylinder number out of range

CD-7, 23, 28 Hard Disk Error Numbers

1	- Drive not ready
2	- Seek timeout
3	- Invalid unit number
4	- Can't find cylinder zero
5	- DMA failed to terminate
6	- No data read
7	- Sector unsafe
8	- Header checksum
9	- Cylinder mismatch
10	- Track mismatch
11	- Sector mismatch
12	- Data field checksum
13	- Status error
14	- Unit out of service
15	- Cylinder number out of range

Network Error Numbers

237	- Hard Disk Busy, Please wait
238	- Output overrun
239	- Semaphore locked/not locked
240	- Relay response timeout
	(FE = Framing, OV = Overrun, PE = Parity)
241	- Data Transmission Error : -- -- FE
242	- Data Transmisson Error : -- OV
243	- Data Transmission Error : -- OV FE
244	- Data Transmission Error : PE -- --
245	- Data Transmission Error : PE -- FE
246	- Data Transmission Error : PE OV --
247	- Data Transmission Error : PE OV FE
248	- Control Block echo comparison
249	- Invalid initial poll command
250	- Data Block input timeout
251	- Control Block input timeout
252	- Incorrect poll code
253	- Incorrect poll responses
254	- Poll timeout
255	- Poll response timeout

BASIC Language Error Messages

Error Code		Meaning
/0	-	Division by zero.
BS	-	Bad Subscript: Index outside DIM statement range.
CN	-	Continue Error: Attempt to inappropriately continue. Can continue from BREAK or STOP if no lines changed or entered. Can't continue after any error.
DD	-	Double Dimension: Variable dimensioned twice. Remember subscripted variables default to dimension 10.
FC	-	Function Call error: Parameter passed to function is out of range.
FS	-	Full Stack: Too many nested FORs and GOSUBs.
ID	-	Illegal Direct: INPUT and DEF statements cannot be used in direct mode.
LS	-	Long String: String too long.
NF	-	NEXT without FOR.
OD	-	Out of Data: More READs than DATA.
OM	-	Out of Memory: Program too big or too many variables.
OV	-	Overflow: Result of calculation too large.
RG	-	RETURN without GOSUB.
SN	-	Syntax error: Typographical error, etc.
SS	-	Semaphore stack overflow.
ST	-	String Temporaries: String expression too complex.
TM	-	Type Mismatch: String mismatched to numeric.
UF	-	Undefined Function: DEF must be executed before function is called.
US	-	Undefined Statement: Attempt to jump to nonexistent line number.

File System Error Codes

128	-	File not found
129	-	Channel not open
130	-	Access Right violation
131	-	Executability violation
132	-	End of file
133	-	Channel already open

SYSTEM PASSWORDS

The system passwords applicable to Keyware OS-65U are listed below. If desired, system security can be maintained by removing this page from the manual and storing separately.

Utility Program Passwords

PASS	All OS-65U programs with limited access (other than Read or Read/Write) are assigned this password. It must be included with the RUN command. PASS is also the password for the OS Keybase data file "CRT0".
PACK	The PACKER program requires this password prior to intiating the disk packing operation.
3300 (CD-36) 3300 (CD-74) 1000 (CD-7)	The COPIER program requires a password prior to initializing a hard disk device. The password is dependent on the system.
UNLOCK	This password is given for the "Enter Immediate Mode" selection on the Main menu.
DISC	The disk configuration program DSKSET asks for this password before allowing execution.

Systems Program Passwords

SECRET	The SYSDIR program password for selection of system 1.
PW	The SYSDIR program password for selection of sample user systems USER1, 2 and 3.
INSTAL	The INSTAL program requires this password before allowing execution.

ASCII CHARACTER CODES

CODE	CHAR	CODE	CHAR	CODE	CHAR
00	NUL	2B	+	56	V
01	SOH	2C	,	57	W
02	STX	2D	-	58	X
03	ETX	2E	.	59	Y
04	EOT	2F	/	5A	Z
05	ENQ	30	0	5B	[
06	ACK	31	1	5C	/
07	BEL	32	2	5D	]
08	BS	33	3	5E	
09	HT	34	4	5F	-
0A	LF	35	5	60	
0B	VT	36	6	61	a
0C	FF	37	7	62	b
0D	CR	38	8	63	c
0E	SO	39	9	64	d
0F	SI	3A	:	65	e
10	DLE	3B	;	66	f
11	DC1	3C	<	67	g
12	DC2	3D	=	68	h
13	DC3	3E	>	69	i
14	DC4	3F	?	6A	j
15	NAK	40	@	6B	k
16	SYN	41	A	6C	l
17	ETB	42	B	6D	m
18	CAN	43	C	6E	n
19	EM	44	D	6F	o
1A	SUB	45	E	70	p
1B	ESC	46	F	71	q
1C	FS	47	G	72	r
1D	GS	48	H	73	s
1E	RS	49	I	74	t
1F	US	4A	J	75	u
20	SP	4B	K	76	v
21	!	4C	L	77	w
22	"	4D	M	78	x
23	#	4E	N	79	y
24	$	4F	0	7A	z
25	%	50	P	7B	{
26	&	51	Q	7C	
27	'	52	R	7D	:
28	(	53	S	73	-
29	)	54	T	7F	DEL
2A	*	55	U		

TERMINAL CODES ENCODING SCHEME

OS-65U supports the commands as listed below on the widest variety of terminals. To adapt the system to your terminal, first run the Terminal Setup program from the Transient Functions menu (/) and check if your terminal is in the list of supported terminals. If it is not, the terminal will need to be added to the list. To accomplish this, you will need the manual for your terminal and OS-65U program ED to edit the CRT command file "CRT 0". Before making any modifications to the "CRT 0" file, it is recommended that you fill out the worksheet at the end of this Appendix according to the instructions below. If you have trouble encoding the commands, contact your dealer or M/A-COM OSI.

For any commands listed on the worksheet that your terminal does not support, enter a zero for the field; note, however, that in such cases the operating system may not perform as documented in this manual.

Supported CRT Commands

The following characters are used by the EDITOR and Extended Input programs documented in the chapter on Transient Functions.

Input Character for Cursor Right
Input Character for Cursor Left
Echo Character(s) for Cursor Right
Echo Character(s) for Cursor Left

The following cursor movement characters are used by the Extended Input program. Note that most cursor movement commands are only available with Extended Input enabled; in order to retrieve these codes, the subroutine found in the "GETCRT" must be run.

Address Cursor
Clear Screen
Clear from Cursor to End of Screen
Clear from Cursor to End of Line
Foreground Follows
Background Follows

Encoding Scheme

In order to install a command set for a new CRT, fill in the CRT ENCODING WORKSHEET for each character according to the instructions below. Enter the codes for each character into the correct field on the worksheet. After the worksheet has been filled in, the codes may be entered into the "CRT 0" file using the instructions provided in Chapter Six of the manual.

Input Character for Cursor Right

Any valid ASCII character is allowed; usually a Control character such as "Control H" is used. Just one character is allowed in this field.

Input Character for Cursor Left

Any valid ASCII character is allowed; usually a control character such as "Control L" is used. Just one character is allowed in this field.

Echo Character(s) for Cursor Right

Upon receipt of the Cursor Right Request character, the operating system will echo up to seven characters. Enter the character or characters that the terminal acknowledges as a Cursor Right Request. At the end of the sequence, enter a decimal zero as a sequence terminator. If this zero is not included, the system will not operate properly.

Echo Character(s) for Cursor Left

Upon receipt of the Cursor Left Request character, the operating system will echo up to seven characters. Enter the character or characters that the terminal acknowledges as a cursor left request. At the end of the sequence, enter a decimal zero as a sequence terminator. If this zero is not included, the system will not operate properly.

Address Cursor

This command is more complex to encode than the others because of the variety of methods an sequences used by terminal manufacturers. By answering the following questions, it should be possible to encode any terminal' s "address the cursor" sequence.

The OS-65U standard procedure assumes that the upper left corner of the the screen is location 0,0 with "x" as the horizontal position pointer and "y" as the vertical position pointer. The Address Cursor subroutine will translate this format to the correct format for the particular terminal.

1) If the terminal requires the vertical position parameter (line) sent before the horizontal parameter (column), enter a "128" in the first byte location for the Address Cursor command on the Worksheet. If the horizontal parameter must be sent first, enter a zero there.

2) Determine the first byte of the Address Cursor leadin command and add its decimal value to the number entered for step 1 (above). For example, if a "128" were entered in step 1 and the first byte of the leadin was an "ESC 27", you would enter 128+27 or "155". This value should replace the one in the first byte position for this command. If the leadin is a multi-byte sequence, enter these values in succeeding locations on the worksheet.

3) If the address cursor sequence does not require a delimiter sequence between the horizontal coordinate and vertical coordinate, write a "0" in the next worksheet locations and skip to instruction 7.

4) If a delimiter is needed, enter a "128" to the decimal value of the delimiter character and enter it in the next byte location for the command.

5) If the Address Cursor sequence does not require a sequence terminator, skip to instruction 7.

6) If the sequence terminator is needed, add "128" to the decimal value of the terminator character and put it on the worksheet.

7) Enter a zero in the next worksheet location.

8) If the horizontal parameter needs an offset from binary zero, enter that offset value in the next worksheet location. If it does not, enter a zero.

   NOTE: Some terminals require an offset of 32 for both parameters. Consult your terminal manual.

9) If the position parameters are to be transmitted as a string of ASCII digits, add "128" to the horizontal parameter offset entered for step 8. If the parameters can be transmitted as single byte binary values, proceed to the next step.

10) If the vertical parameter needs an offset from binary zero, enter that offset value in the next location. If it does not, enter a "0".

Clear Screen

Enter the character sequence required to clear the terminal screen. Add a zero terminator at the end of this sequence. It is assumed that the terminal will "home" the cursor before clearing the screen. If your particular terminal does not do this, then the command sequence must include a "home cursor" request before the "clear screen" request.

Clear from Cursor to End of Screen

Enter the character sequence required to clear from the cursor to the end of the screen. Include a zero terminator at the end of the sequence.

Clear from Cursor to End of Line

Enter the character sequence required to clear from the cursor to the end of the line. Include a zero terminator at the end of the sequence.

Foreground Follows

Enter the character sequence to highlight or underline the characters that follow. Include a zero terminator at the end of the sequence.

Background Follows

Enter the character sequence to switch back to normal intensity or mode. Include a zero terminator at the end of the sequence.

Use the instructions given in the "Terminal Setup" section of Chapter Six of the manual to enter the terminal codes in the "CRT 0" file.

CRT ENCODING WORKSHEET

Cursor right Input

Cursor Left Input

Cursor Right Echo

Cursor Left Echo

Address Cursor

Clear Screen

Clear to End of Screen

Clear to End of Line

Foreground Follows

Background Follows

OS-65U VERSION 1.43/1.44 ENHANCEMENTS

This appendix describes the changes that have taken place in OS-65U in Version 1.43 and 1.44.

OS-65U VERSION 1.44

There are three changes between OS-65U Version 1.43 and 1.44: The name of the company has been changed to Ohio Scientific; the Disk Systems Manager has been added to the OS-65U software package, and a new version of the Networking Timesharing and Print Spooling documentation has been included in the basic OS-65U Reference Manual.

OS-65U VERSION 1.43

Version 1.43 enhancements include bug fixes, improved utility programs, and better hardware adaptability. We wish to encourage reporting of any bugs or desired system enhancements to Ohio Scientific Software Development, 7 Oak Park, Bedford, MA 01730.

The changes in Version 1.43 are summarized below. Note that some of the changes are global in nature. Avoid using old versions of system utilities or overlays if a new version of the program is present on the new system.

OS-65U Version 1.43 is available for the CD-36/74, CD-28 and CD-7 hard disk based computers in single user and time share configurations. Networking configurations are available for the CD-36/74 and CD-28 systems.

Time share software is now standard when OS-65U is purchased, irrespective of whether the computer hardware is capable of time share operation.

A print spooling package is standard with OS-65U. Refer to the chapter on print spooling for details of its operation.

Two new FLAGs have been added to Version 1.43. FLAGs 31 & 32 enable & disable auto-lower to upper case translation on line input. FLAGs 33 & 34 modify the LIST command to print all characters lower case except for data within quotes and REM statements. Tokens are printed in lower case with the first letter capitalized. See the FLAG section for a more detailed description.

A new parallel word processing printer driver WPDRIV has been added to Version 1.43. This driver replaces TECDRV and NECDRV which were incompatible with Version 1.30. This driver may be enabled and disabled. Note that this utility works only on serial-based systems.

The execution of the utility programs DELETE, RENAME, and COPYFI has been made much faster.

The PACKER program formerly contained a SN error when the DIREC* file was longer than one sector. It also kept the floppy diskette head loaded when encountering a disk error. These bugs have been fixed, and the program has been made easier to ABORT.

The CHANGE program formerly contained a bug in that negative offsets would result in disk write output always being directed to DEVice "A". This problem has been eliminated in Version 1.43.

The COPIER program's DEVice specification checks were improved for the intelligent workstation environment of the network.

The RENAME program formerly did not offer the option of changing the file's access rights. It now offers that option.

The "CRT 0" file has been simplified so that the first record in the file is the default for the system.

The PRTMAP program was added; it is a general purpose logical to physical printer mapping program that permits DEVice #5 to be routed to the physical drivers for DEVices 3, 5, 6 or 8. It also optionally sets system paging parameters. If paging is enabled and the printer accepts a CHR$(12) for top of form, the system top of form commands (FLAG 100 & FLAG 101) will be modified to use it. These options are selected by answering screen prompts.

A system configuration program (PRTSET) has been added that permits DEVice "3" and DEVice "8" serial input and output drivers to be directed to any free ACIA port(s) in the computer. The standard port driver addresses have been changed. DEVice "3", which was set to $FB00 before, is now set to $CF02. DEVice "8" is set at $CF00 as before. These addresses can be changed by using the program PRTSET.

The DEVice "8" I/O driver was rewritten to be a standard index driven serial input driver. The default port offset in the DEVice "8" driver was changed from 255 to 0.

The time share programs (TS3674, TSCD23, TSCD07) will auto-configure themselves to the old or new hardware (C3 vs C200-C300 series). The time share executive has been modified to increase system throughput.

BASIC's internal rounding was turned back on. It had been disabled in Version 1.30 when Extended Input was introduced. To disable rounding, refer to a listing of INP$ for the POKE which has been REM'd out.

A file I/O bug has been corrected which resulted in a small speed improvement during file I/O operations.

The network scheduler was modified to service network intelligent terminals equally.

In earlier network releases, it was necessary to wait one second before a retry on a time limited WAIT FOR or WAIT CLEAR from an intelligent terminal. This retry rate has been reduced to 50 milliseconds.

If a user does a time limited WAIT FOR, and that user has already set that semaphore, the system will indicate that the user has successfully locked that semaphore. NOTE: This indication is given only with time limited WAITs. A WAIT for ever command will still result in a wait for ever.

The hard disk Get/Put driver had a bug which resulted in an occasional sector being written to disk without first reading the sector. This has been corrected.

The Transient Functions have a system resident indicator byte that has the currently enabled Transient and the requested Transient. This greatly simplified the enablement and disablement logic for the Transients. Also, the Transient Function programs have user-definable entry points in the 63500 line number range which can be used to facilitate system customization with respect to these programs.