DEALER UPDATE ON TWINSYSTEM DOCUMENTATION

The TwinSystem is a 2-user version of the System 88. The TwinSystem operating system arbitrates allocation of devices (disks, printers, programs) between the two users. These changes have resulted in differences in system use that are perceived by the user as well as differences which must be dealt with by the applications programmer.

Attached is the preliminary documentation on the TwinSystem 88.

Section 1 of this preliminary document instructs the end user on the operation of the TwinSystem. It is intended that this document be accompanied by the System 88 User's Manual. Then, as soon as possible, we will make this section a part of the user's manual, and TwinSystem owners can receive replacement manuals. When we reprint the user's manual we will include this TwinSystem documentation, as well as make special notes to TwinSystem users at any point in the user's manual where there are differences between single and two-user systems. We emphasize that these differences are few and that the following section should be sufficient to allow the successful use of this new product until all of the changes and additions have been made to the user's manual.

The second section in this document is pertinent to dealers and applications programmers. It explains how to adapt existing BASIC and Assembly programs for successful use on the TwinSystem and contains a dealer-oriented explanation of how the TwinSystem functions. Also included with this preliminary document is the addendum to the BASIC manual which reflects recent changes to BASIC. Some of the changes to BASIC are bug fixes and some were necessitated by TwinSystem 88 development.

In addition to our work on a revised user's manual, we are preparing a revision of the System 88 System Programmer's Guide. This manual is being revised so that it can be used by system and applications programmers who are writing programs for the TwinSystem. In the meantime, the developers of the TwinSystem can be contacted directly about difficulties encountered by programmers who are adapting or writing software for the TwinSystem.

Please direct written suggestions and complaints concerning end-user documentation to Jennifer Douglas. Bob Martin will receive written suggestions for the changes that need to be made to the System Programmer's Guide. Patty Ryan is prepared to help with problems over the phone, but it is important that written suggestions and complaints be received by Bob Martin and Jennifer Douglas so that we can make the documentation suitable for your use.

This manual is part number 81Ø18Ø. It is the Preliminary TwinSystem 88 Documentation that is to be replaced by the revised System 88 User's Manual and the revised System Programmer's Guide. This documentation describes the revision to the System 88 Operating System to allow two-user support. This revision was done by R. T. Martin. This documentation was written by Jennifer Douglas and R. T. Martin.

Section 1

TwinSystem Operating Instructions

The TwinSystem allows two users to share the same System 88. Each user has a separate 48k of memory space and a separate keyboard and display. The two users share the central processor, the disk drives, and the printer driver. The TwinSystem Operating System contols and arbitrates access to the shared resources on a first-come-first-serve basis, thereby protecting both users from starting a function they cannot safely complete.

1.1 LOADING THE SYSTEM

When a TwinSystem disk is inserted in one of the TwinSystem drives, and the TwinSystem "Load" button is pushed, both displays will show the Exec $ prompt.

The "System Residence Drive" (drive 1 unless an 88/MS is attached in which case it is drive 4) is the drive which the system checks first when the load button is pushed. If there is a system disk (a disk with the System 88 Operating System on it) in the system residence drive, then that is the disk that the system loads into memory. If there is no system disk in the system residence drive, then the system checks each drive (in drive number order) until it finds a drive with a system disk.

> NOTE: Typing on either keyboard before the system is up and has displayed text on both screens may cause the system to hang, i.e. not complete the boot process. If this mistake is made, push the Load button again.

In order to discover which user is user-1 and which is user-2, type **ENABLE** (either or both users can do this.). The result is that the Exec $ prompt changes to a **1$** on the user-1 side and a **2$** prompt on the user-2 side. It is helpful to note who is user-1 and who is user-2.

1.1.1 Using INITIAL File Capability

The TwinSystem 88 has the two-user version of the **INITIAL** file capability. If both users are sharing the same operating system disk, each can arrange to have a .GO program (an application) loaded automatically when the system is loaded. To arrange for this convenience, rename the .GO file that user-1 needs to have loaded automatically, **INITIAL1**, and rename the .GO program that user-2 needs loaded automatically to **INITIAL2**.

Then when the system is loaded the two sides of the system will automatically begin running their respective initial programs.

If it is desired that both users run the same application program, and that program is to be started up automatically when the system is loaded, then it can be renamed **INITIAL.**

If both users are using separate system disks (the procedure that causes the system to load a second system disk into memory and then run the second user with it is described later) they can each rename a .GO program to **INITIAL** and their respective programs will run automatically when their system disks are loaded into memory.

1.1.2 Using Single-user Software on the TwinSystem

Single-user System 88 software can be used on the TwinSystem 88. There are special procedures for doing so and they are explained in detail at the end of this section.

> **WARNING:** Do not attempt to run any single-user program or system disk before reading 1.6 of this document.

1.2 POSSIBLE TWINSYSTEM 88 CONFIGURATIONS

The TwinSystem 88 has several possible configurations.

The minimum recommended configuration is a 3-drive double density 8813.

Alternative recommended configurations are a 1, 2 or 3 drive 8813 with a 2-drive 88/MS add-on storage unit.

The maximum alternative configuration is a 1, 2 or 3-drive 8813 with two 2-drive 88/MS add-on storage units.

1.2.1 Optimum Operating Procedure for your Configuration

There are different recommended procedures for using different configurations of the TwinSystem to their best advantage. Soon we explain the first-come-first-serve arbitration procedure. You will see that it is most efficient if each user is assigned their own mass storage disk (whether it be a 5" DD disk or an 8" DD disk. This is a recommended procedure for all possible configurations as it protects each user from delays caused by two users needing to access the same drive at the same time. In addition to the recommendation that each user be assigned his own mass storage disk, there is an additional arrangement that should be made if the system being used has only 2 drives. We explain this below.

1.2.1.1 2-Drive TwinSystem 88

If the TwinSystem only has 2 drives, then at least one of the two drives has to contain a disk which has a dual function. This disk must serve as the storage disk for one user and as the

system disk for one or both. As you will soon learn, there is one function (**PACK**) which when done on a shared operating system disk, is disableing both users for its duration. To avoid this interruption, both users can agree to use separate operating system disks. This can be done by placing an operating system disk in each of the two drives. When the system is booted, it automaticaly loads the contents of the system disk in the system residence drive. If user-1 has been assigned the system residence drive, then user-2 can use the **boot n** command to cause his own operating system to take over the running of his side of the system. The **boot** command only works when the system is in the enabled mode. The second user can enable his own side of the system independently of the other user and then enter the **boot** command followed by the drive specifier of the drive he is using for his combined operating system and mass storage disk.

This means that user-2 must type the following after the 2$ prompt:

> **boot n** (where n is the number of the drive containing his operating system and mass storage disk)

1.2.1.2 3-drive TwinSystem 88

If your configuration has three drives or more, there is no problem with users sharing one operating system disk, and, this configuration allows each user to have a disk which is completely devoted to mass storage.

1.3 ALLOCATION OF SHARED DEVICES

There are primarily two types of devices that are shared, the printer and the disk drives. If the printer is being used, or if any printer driver-related function is being performed by one user, the other user can not perform any printer driver-related function. Allocation of drive privileges is not as clear cut however. Both users can share the same drive at the same time if they are performing procedures which do not conflict. We have already suggested that each user be assigned his own drive to minimize conflicts that result when two users request privileges on the same drive at the same time. Even though this is a recommended procedure, it is good that you understand how the TwinSystem arbitrates requests for device privileges. There may be times when both users will need to share the same drive, as when both are sharing the same data or text files, or when both are running the same application. This section explains the arbitration and granting of privileges that occurs automatically when a System 88 command is issued.

When a user types a system command, the system performs some action. If that action involves the use of a shared resource, the system must determine two things:

1. it must decide if the shared resource is in use

2. if the shared resource is in use, it must determine what action the other user is performing

Some actions can occur and successfully finish on the same device at the same time. The system allows those actions to occur simultaneously. Other actions preclude some or all actions that the other user might want to do on the same device. In those cases, the system denies immediate access to the shared resource and displays the message:

That device is busy.

The user need not be concerned with this arbitration process; it is automatic. In fact, the use of the system commands is nearly identical to that described in the System 88 Exec Commands section. However, it is helpful to understand that implicit in any system command given on a TwinSystem is the request to use the named device for a particular action. If that action cannot be guaranteed to be successful when done simultaneously with an action already in progress, permission is denied.

To determine whether or not a particular action is permitted on a particular device, the system categorizes the types of actions and always remembers what types of actions are already occuring on which devices. The categories it uses are: read, write, dirmod, and exclusive. Each command implies one of these types of actions as shown in the following chart:

READ	DIRMOD	WRITE	EXCLUSIVE
LIST	RENAME	BASIC INOUT FILES	PACK
TYPE	DELETE		IMAGE
	UNDELETE		INIT
	COPY		Printer SET
	EDIT		Printer SHOW
			Printer printername
			PRINT
			FORMAT
			CHECKSUM

The following chart shows which actions can take place on the same device at the same time, and which actions must be arbritrated in the first-come-first-serve manner. It shows, for example, that when one user has dirmod privileges, the other user can be granted read or write priveledge but not dirmod or exclusive.

	read	dirmod	write	exclusive
read	both	both	both	1st
dirmod	both	1st	both	1st
write	both	both	both	1st
exclusive	1st	1st	1st	1st

In summary, there are a few characteristics of this resource allocation system that should be kept in mind.

Two functions initiated by the two users which require modification of a directory on the same disk will cause the second user to see the **"That device is busy."** message.

Use of the **INIT** command on an 8" drive will prohibit access by the other user to any 8" drive in the system until the **INIT** procedure is completed. If some procedure is in progress that involves access to an 8" disk, access is held up until the **INIT** procedure is complete.

A **PACK** of the system disk can not occur unless the system is set in the **SOLO** mode. The **SOLO** mode can only be invoked when the other user is in Exec; this protects the second user from being interrupted by the **PACK** function, and allows the **PACK** procedure to complete successfully. See the following section on the use of the **SET** command with all of its possible arguments.

1.4 USING THE SET COMMAND

The standard mode of operation is to arbitrate shared resources on a first come-first serve basis. Ordinarily, when a user is in the Exec mode, he has no priority over any shared device. This means that one user can possibly gain privileges for a device which the other user has been using more or less continuously but has temporarily given up to return to Exec. For instance, one user might be printing a long document. Perhaps a part of the document has been printed and the user is proofing it and preparing to re-print. During this short interval the printer is available on a first-come-first-serve basis. If the other user issues a command that implies a request for printer privileges, those privileges are granted as long as the other user is not using the printer at the moment. At times, this

first-come-first-serve procedure can be an inconvenience. To avoid this there is an alternative means of allocating privileges. The **SET** command can be used to reserve a device for a particular function, suspending the first-come-first-serve procedure until the **SET** command is cleared.

The **SET** command is always issued from Exec and usually includes a device name (printer) or drive number (1 or 4 or 5) and one of the classifications of functions (dirmod, read, write or exclusive.) For example, the following command requests uninterrupted use of the printer until the **SET clear** command is used.

SET printer exclusive

The only function type that makes sense when used in connection with printer functions is exclusive since the printer can never be shared. The drives can be shared, however, so there are several possible arguments that can be given with the **SET** command when privileges for a drive are desired. The chart in the preceding section shows that two users can be granted write privileges at the same time, whereas two users cannot be granted dirmod privileges at the same time. This chart applies to the granting of uninterrupted privileges as well.

The **SET** command and all of its possible arguments are as follows:

SET printer exclusive
SET n exclusive
SET n dirmod
SET n write
SET n read
SET n read and write
SET n read and write and dirmod
SET n read and dirmod
SET n write and dirmod

n is the drive number of the shared device.

To clear any of the above **SET** command forms, use the following format:

SET n clear

Two additional **SET** commands allow you to intialize a shared system disk. The **SET SOLO** and **SET TWIN** commands do not require device name or numbers because they affect the entire system. In order to pack a shared system disk, the **SET SOLO** command must be issued. This command must be issued when both users are in Exec. The **SET SOLO** command results in the non-requesting user being disabled until the **SET TWIN** command is issued.

1.5 SCHEDULING OF IMAGE AND INIT FUNCTIONS

Since the **INIT** process performed on an 8" drive delays 8" drive access for the other user, the **INIT** procedure should be done while the other user is not at the system. It is best to have some initialized disks standing by so that you do not need to worry about planning for initialization time.

The **IMAGE** procedure should also be done while the other user is not at the system as it requires the use of two drives. Perhaps a particular period of time can be set aside for backing up disks, as this procedure takes a few minutes if 8" drives are involved.

1.6 USE OF THE TWINSYSTEM AS A SINGLE-USER SYSTEM

If there is a need to run old single-user programs on the TwinSystem, the following steps must be followed.

1. Insert the TwinSystem System Disk. Push the Load button. When the Exec prompt appears on the screen, type **ENABLE.** Either a 1$ or a 2$ prompt will replace the $ prompt. The 1$ prompt indicates that the screen on which it appears belongs to user 1. Having determined which display and which keyboard belong to user 1, disconnect the user 2 keyboard.

2. Remove the TwinSystem System Disk.

3. Insert the old single-user system disk and push the Load button.

4. The TwinSystem is now functioning as a single-user system, and the old single-user system disk can be used exactly like it is used on a single-user System 88.

5. It is extremely important that the second user's keyboard be detached from the TwinSystem while it is being run as a single-user system. Any input from the second user can cause the single-user operation to fail.

   WARNING: DO NOT ATTEMPT TO RUN A SINGLE-USER OBJECT FILE WHILE THE TWINSYSTEM IS BEING OPERATED IN THE TWINSYSTEM MODE. If the TwinSystem is being run as a TwinSystem and one of the two users inserts a single-user system disk and attempts to use the boot command to allow the drive to operate independently of the TwinSystem System Disk, the result will be a catastrophic system failure or the following error message will appear on the errant user's screen:

   Single-User Trap--- Reboot!

Attempting to run any single-user object file on any disk in any drive while the TwinSystem is being run as a TwinSystem, results in the same catastrophic result or error message.

After the above described mistake has been made, the system must be rebooted, as the system assumes that the data structures for the user that ran the single-user program are destroyed.

Section 2

ADDITIONAL TECHNICAL INFORMATION FOR PROGRAMMERS AND DEALERS (NOT FOR END-USERS)

NOTE: In the absence of the new System 88 Programmer's Guide we include this section which is a collection of information intended for the programmer who needs to convert his BASIC programs to run on the TwinSystem, and for the dealer who wants some general information about the TwinSystem.

2.1 RUNNING BASIC PROGRAMS

BASIC programs that run on existing single-user software will run unchanged on The TwinSystem if they meet the following conditions:

1) Run under COlL BASIC in 48k or less
2) Do not use CALL
3) Do not use PEEK and POKE other than to the contents of the video display

If a program does not run in 48k with COlL BASIC, it must be modified to run in 48k or less. The "large" flavor of BASIC will be the only one offered on the TwinSystem. Programs that use PEEK and POKE to other than the video display are highly discouraged. Most everything except video display has moved in memory between the single and two user system. This also makes programs that make use of CALL require tinkering.

2.2 RUNNING ASSEMBLY PROGRAMS

NOTE: Because of the high potential for chaos caused by undebugged assembly code programs running on the TwinSystem, an assembler is not part of the first release. The assembler will be available with the revised System Programmer's Guide, which discusses the nitty gritty of TwinSystem operation.

A berserk assembly code program can cause a great deal of grief in the TwinSystem. Most everything has moved in memory between the single and the two-user system (i.e. wormholes, system service vectors, etc.) The system itself now lives in RAM that cannot be protected from errant stores. If a program crashes the system while the other user is packing a disk, some or all the data on the disk will be lost.

In general, assembly programs that use the REFS and REF

statements for linkage with system routines and data will require very little tinkering. Programs must respect MEMTOP and not disable interrupts for long periods of time. Programs that wait for an event by executing an EI/HLT pair, should replace the EI/HLT with a call to a system routine to give up the processor. Details will be given in the System Programmer's Guide and are also available from the Guide's originator, R. T. Martin.

2.3 CUSTOM PRINTER DRIVERS

The printer driver has moved from its normal memory position in the single-user system. This move was necessitated by the interlocks required in two-user operation, as well as other considerations. The lack of an assembler for the first release also makes the entire question moot. Those requiring translation of existing custom drivers should contact R. T. Martin at PolyMorphic Systems.

2.4 SSSD SUPPORT

It was stated with the announcement of the TwinSystem that SSSD disks are not supported by TwinSystem software. This is because of the servere performance problems involved with programmed data transfers to and from SSSD disks. The best way to handle this problem is to transfer (or image) SSSD files and/or disks to either DSDD 5" media or to 8" media for use on the TwinSystem.

2.5 HOW TWINSYSTEM WORKS - BRIEFLY

In TwinSystem hardware each user has a video board and a 48k memory board. The system has an 8k memory board for itself. The 8k board is addressed to E000H. Both video boards appear at 1800H, and both 48k boards appear at 2000H. The CPU board is modified to control the PHANTOM line on the S-100 bus. The 48k and video boards are modified to recognize this PHANTOM line. One video/48k board pair is active when PHANTOM is high, and the other video/48k board pair is active (selected) when PHANTOM is low. So, each user has a separate memory space and video interface. The memory layout for each user is the same. Each has a separate video display, overlay area, directory area, user area, and stack. The user stack is kept at the top of the 48k board, starting at location E000H and working down. 512 bytes is allocated for stack space, putting top of memory (MEMTOP) at DDFFH.

To switch between users, the system pushes the user's registers into the stack and stores the stack pointer in the user space. Then the system switches the PHANTOM line to deselect the old user and select the new one. The stack pointer is reloaded, the user's register's popped from the stack, and away we go.

Control switches between the users in one of two ways. The first is the 60 cycle clock. Every 1/60 of a second, when the clock

interrupts the processor, the system parks the current user and switches to the other user. For this to happen, interrupts must be disabled, and a few other conditions must be met. The second way in which control switches between the two users is that one of them gives up the processor. When one user is waiting for something (e.g. a character from the keyboard or a disk transfer to complete), that user is still run at least 60 times a second. But while waiting, that user just checks to see if the wait condition is satisfied and gives up the processor if it is not.

There are a number of fairly significant complications in disk handlers with regards to two-user operation, e.g. the interlock mechanism that protects a user from initializing a disk that the other user is packing at the time.

2.6 KEYBOARD INTERFERENCE DURING HEAVY DISK I/O

If one side of the system is peforming very heavy disk I/O, the other user may experience some difficulty when touch typing. The cursor may seem to freeze up for a second or two, and thus interfere with rapid, smooth typing, possibly causing some mis-spellings because some characters which are typed are not actually processed. Take extra care during those periods when you detect that the cursor is freezing in place for a fraction of a second or so.

2.7 NEW UNDOCUMENTED COMMANDS

This would not be a new system without new undocumented commands. Some will stay undocumented for quite a while. To observe the other user's program counter and registers: Type PCmon in enabled mode; this is exited by control-Y. To watch the device interlock code work: Type SET YAK ON PLEASE in either enabled or disabled mode. SET YAK OFF PLEASE disables the messages from the device interlock code. This "feature" will probably vanish when space is needed. The SET code also folds all its input to upper case, so that YAK yak Yak yAk are all equivalent. SET also understands spelled-out numbers from 1 through 9, so that "SET 4 EXCLUSIVE" and "set four exclusive please" are equivalent. As you may have guessed, PLEASE is a buzzword that is ignored.

2.8 NEXT RELEASE WISH LIST

Add to this section the things that you think you want or need. They just might get implemented.

DC Hayes support for one or both users on modem
One user running over serial port
More efficient I/O code

BASIC CØ2 Features

PEEK, POKE, and CALL

In order to provide compatibility in BASIC programs between single and two-user systems, PEEK, POKE, and CALL functions on addresses from ØØ to 255 inclusive have been changed.

1.1 PEEK

The PEEK function returns as its value the contents of the memory location selected by the argument to the function. Z=PEEK(43) returns the contents of location 43, which is assigned to the variable Z. Since memory locations Ø through 3Ø71 are read only memory in the system, other functions have been assigned to them. Note that some of the functions or locations provided are only useful in coordination with the information contained in the System Programmer's Guide. Note that these special PEEK functions are supported only by BASIC version CØ2 and later.

1.1.1. PEEK(Ø)

Returns the 16 bit cursor address from the system cell POS. Together with POKE Ø, may be used to check and alter the cursor character on the video display.

1.1.2 PEEK(1)

Returns the 8 bit contents of system cell ONCE. This gives state information to the program. See System Programmer's Guide for details.

1.1.3 PEEK(2)

Returns the address of the character input wormhole, WHØ. The addresses of the other wormholes may be calculated from this address, as each wormhole vector is 4 bytes. So, the address of WH1 is 4+PEEK(2).

1.1.4 PEEK(3)

Returns the 16 bit address of the system command buffer, CMND.

1.1.5 PEEK(4)

Returns the 16 bit system command pointer, CMPTR. This points into the system command buffer, and may be used to retrieve arguments from the command line.

1.1.6 PEEK(5)

Returns a Ø on a single-user system, and 1 or 2 on a two-user system, depending on user number. This is used by a program to decide if it is running on a single or two user system, and if on a two-user system, which user it is.

1.1.7 PEEK(6)

Returns the eight bit value of the screen home address, SCRHM.

1.1.8 PEEK(7)

Returns the eight bit value of the screen end address, SCEND.

1.1.9 PEEK(8)

Returns the version number of BASIC. Note that this may be used by a program to see if it is running on CØ2 BASIC, or an earlier version. Earlier versions of BASIC will return the eight bit value of memory location 8, which is 245 (ØF5 hexadecimal). BASIC version CØ2 returns CØ2 in decimal, which is 3Ø74.

1.2 POKE

The poke statement is used to alter memory locations. This statement should always be used with care, as it may cause system failure, which can be especially hazardous on a two-user system. The actions of the POKE statement for some addresses between Ø and 7 have been redefined.

1.2.1 POKE Ø,X

Stores the character with decimal value X at the current cursor position. This is useful for erasing the cursor from the screen. Remember, when putting characters on the video display in this manner, that the normal ASCII code must have 128 (decimal) added to it, or the code will be displayed as a graphics character. So, to put a space character (2Ø hex, 32 decimal) at the current cursor position, the statement POKE Ø,128+32 would be executed.

1.2.2 POKE 1,X

Stores the byte value (Ø<=X<=255) into system cell ONCE. See System Programmer's Guide for details.

1.2.3 POKE 6,X

Stores the byte value into system cell SCRHM, the screen home address.

1.2.4 POKE 7,X

Stores the byte value into system cell SCEND, the screen end address.

1.3 CALL

The CALL function is used in BASIC to call assembly language routines and overlays. The value returned by the CALL function is the 16 bit result left in registers HL by the assembly language function invoked. CALL to locations Ø, 1, and 2 have been redefined as follows.

1.3.1 CALL(Ø

Forces all file channels closed and exits the program gracefully to the system Exec.

1.3.2 CALL(1

Calls the system service routine Dio. See System Programmer's Guide for details. Note that an error return from Dio will be taken as a BASIC error and handled accordingly.

1.3.3 CALL(2

On the single-user system, returns with no change. On the two-user system, calls the system service routine Devlock. See System Programmer's Guide for details.

1.3.4 CALL("Berr",5,Ø,X,Ø)

This calls the Berr overlay with error code X and causes the overlay to print the corresponding error message on the screen. Note the message is NOT terminated by a carriage return. For example, the following BASIC code will print the line number of the error and the error message as trapped by an ON ERROR statement:

```
1ØØ PRINT "Line",LINE," reports ",Z=CALL("Berr",5,Ø,ERR,Ø)
```

If LINE has the value 5ØØ and ERR has the decimal value 1Ø25, the text

Line 5ØØ reports Syntax error.

would be displayed on the screen.