RAMDISK

REFERENCE MANUAL

BUD MILLS SERVICES
166 DARTMOUTH DR.
TOLEDO, OHIO 43614

HORIZON COMPUTER, LIMITED

PO BOX 554 WALBRIDGE OHIO 43465

EXCLUSION OF WARRANTIES

The information contained in this manual is subject to change without notice. Horizon Computer, Limited shall not be liable for technical or editorial errors or omissions contained herein; nor for incidental or consequential damages resulting from the furnishing, performance, or use of this material or product described by this material.

Horizon Computer, Limited makes no warranty, expressed or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose. All software, hardware, and printed materials are made available solely on an AS IS basis.

THE USER ASSUMES SOLE RESPONSIBILITY FOR DETERMINATION OF FITNESS OF THIS PRODUCT FOR A PARTICULAR APPLICATION.

LIMITED THREE MONTH WARRANTY

Horizon Computer, Limited warrants its fully built and tested RAMDISK to be free from defects in materials and workmanship for a period of three months from the original date of purchase. This warranty shall extend only to the original consumer purchaser. This warranty is void if the RAMDISK has been damaged by accident, unreasonable use, neglect, improper service, or other cause not arising out of defects in materials or workmanship.

No warranty is made for products other than completed RAMDISKs, including printed circuit boards and other parts, software programs, and printed materials.

Horizon Computer, Limited shall not be liable for loss of use of the RAMDISK or other incidental or consequential damages incurred by the consumer or any other user. In any case, Horizon Computer, Limited shall be liable only for the cost of the RAMDISK and associated materials.

Users Manual by D. R. Romer
Construction Guide by R. Gries
ROS modifications by J. Peter Hoddie

COPYRIGHT 1986, Horizon Computer, Limited

ALL RIGHTS RESERVED

TABLE OF CONTENTS

INTRODUCTION 1

COMPATIBILITY 1

INSTALLATION INSTRUCTIONS 2

- Unpacking the Card 2
- Determining CRU Base Address 2
- Installing the Card 4
- Loading the RAMDISK Operating System 4
- Loading DM-1000 Disk Manager 6

CALL SUBPROGRAMS 7

- CALL DN(n) Set Drive Number 8
- CALL MS(n) Set Max. Sectors 9
- CALL WO Turn On Write Protect 9
- CALL WF Turn Off Write Protect 9
- CALL EX(a) Execute Machine Language 9
- CALL CO Enable DSR RAM 10
- CALL CF Turn Off DSR RAM 10
- CALL DM Load DM-1000 from RAMDISK 10
- CALL NF(n) Set Number of Floppy Drives 10

DELETE "XBCALL" Download Ext. BASIC Routines 11

COMMON RAMDISK APPLICATIONS 12

- TI Writer / TK Writer 12
- TI Multiplan 12
- TI BASIC and TI Extended BASIC 13
- Editor/Assembler 13

IN CASE OF DIFFICULTY 14

- Testing for Repeatability 14
- RAMDISK as Cause 14
- Testing the Memory with MEMTEST 15
- Inability to Access Floppies 15
- Lock-up on Power-Up 16

ADVANCED APPLICATIONS 18

- Introduction 18
- Card Architecture 18
- How to Enable the Memory 20
- Loading Your Own Programs 21
- Experimenting with DSR's 23
- Writing CALL Subprograms 23
- Disk Contents 28

RAMDISK CONSTRUCTION GUIDE 30

- Introduction 31
- Eight Construction Steps 32
- Schematic and Parts List 39

INTRODUCTION

The HORIZON RAMDISK is a peripheral card for the TI-99/4A Home Computer. The RAMDISK is a battery-backed, solid state device which is intended to emulate all functions of a TI floppy disk drive. The primary advantage of the RAMDISK is speed; data transfer takes place roughly twenty times faster.

To use the HORIZON RAMDISK you need:

1) TI-99/4A console and TV or monitor

2) 99/4 Peripheral Expansion Box

3) TI Compatible Disk Controller Card

4) At least one floppy disk drive

5) One of the following Command Modules

   A) TI Extended BASIC

   B) Editor/Assembler

   C) TI-Writer

6) 32K Memory Expansion (recommended)

COMPATIBILITY

Every attempt has been made to insure that the HORIZON RAMDISK will be compatible with TI and third party software. The RAMDISK appears to be compatible with TI Writer, Editor/Assembler, Multiplan, Logo, Forth, BASIC, Extended BASIC and Assembly Language. Most applications concerning the above should be compatible with the RAMDISK. However, some software, designed to directly access the Disk Controller Card will not recognize the RAMDISK and hence is not compatible. Examples include the CorComp Disk Manager Program and TI Pascal. Non-standard disk access techniques or disk protection schemes could also cause compatibility problems (e.g., TI-ARTIST).

HORIZON RAMDISK

INSTALLATION INSTRUCTIONS

Horizon Computer, Limited

```
*****************************************
*                                       *
* CAUTION: Your HORIZON RAMDISK contains *
*                                       *
* components which can be damaged by    *
*                                       *
* STATIC ELECTRICITY. Please follow the *
*                                       *
* instructions below CAREFULLY!         *
*                                       *
*****************************************
```

1) UNPACKING THE CARD

BEFORE UNPACKING YOUR CARD... you should know it contains CMOS RAM "chips" which are damaged by STATIC ELECTRICITY MORE EASILY THAN OTHER LOGIC TYPES. Therefore, you should take proper precautions in handling the card as described below. THIS IS EXPECIALLY TRUE IN LOW HUMIDITY ENVIRONMENTS -- REMEMBER THAT WINTER PRODUCES IDEAL CONDITIONS FOR STATIC!

Most people are aware of the kinds of things that produce static charges -- synthetic garments, carpeting, plastics (not specially treated), etc. When handling the card, please use common sense and AVOID CONDITIONS WHICH MIGHT PRODUCE STATIC CHARGES.

Your card has been packaged in a pink static protective bag. Save this bag and use it when you transport the card or if you have to send it back for service.

Before unpacking the card, take appropriate precautions to prevent static charges. Then make sure you ground yourself by touching a metal object -- the INSIDE METAL structure of your Peripheral Expansion Box is ideal. When handling, try to touch ONLY the edges of the card, and DO NOT touch the CMOS memory chips themselves -- they are the larger devices which occupy most of the board.

2) DETERMINING THE CRU BASE ADDRESS

Your HORIZON RAMDISK has been provided with a DIP (Dual In-line Package) switch for setting the

base CRU address the computer will use to access the RAMDISK. Look at the top side of the card near the area where the edgecard connectors are located. The DIP switch is labeled "switch" and is located between U19 and U20.

During construction, this DIP switch should have been set so that the RAMDISK will respond to CRU address >1000 -- the top switch (1) should be toward the "ON" or "CLOSED" position, and all other switches should be toward the "OFF" or "OPEN" position.

Setting the CRU base address to >1000 insures that your RAMDISK will be the first device checked on all Input/Output operations. IT IS THE BEST WAY TO SET YOUR CARD FOR NEARLY ALL COMMON RAMDISK APPLICATIONS (e.g., TI Writer, Editor/Assembler, Multiplan, BASIC, TI FORTH, etc.)

However, under special circumstances, it may be necessary to change the DIP switch setting. If you have another device in your system which is known to be using CRU address >1000 (such as a MYARC or other brand ramdisk), or if you wish to use a second HORIZON card (which, as will become apparent later, can be much more than a RAMDISK), then you may need to change your DIP switch setting. Just remember that ONLY ONE switch should be set in the "ON" position and that, as shown below, some positions are already in use -- DO NOT USE ANY CRU ADDRESS ALREADY IN USE IN YOUR SYSTEM!

DIP Switch	CRU Address	Normal Use
1	>1000	Not normally used
2	>1100	Disk Controller Card
3	>1200	for Modem Card/not used
4	>1300	RS232 Ports 1 and 2
5	>1400	Unassigned
6	>1500	RS232 Ports 3 and 4
7	>1600	Unassigned
8	>1700	Hex Bus/not used

If you should decide to change your CRU base address from >1000 then DSK1, DSK2, DSK3, and DSK4 (if you have a CorComp disk controller) will be unavailable to the RAMDISK -- the disk controller card at >1100 will always be checked before the RAMDISK and the routines on that card will supplant the corresponding RAMDISK routines.

So, in summary, we recommend that unless you have a strong reason to change it, leave DIP switch 1 "ON" so the CRU base address of the card will be >1000.

3) INSTALLING THE CARD

```
*****************************************
*                                       *
* CAUTION: As with ALL Peripheral Expansion *
*                                       *
* Box (PEB) cards, you MUST turn OFF PE-Box *
*                                       *
* power and wait a FULL TWO MINUTES before  *
*                                       *
* installing or removing the RAMDISK card.  *
*                                       *
*****************************************
```

After removing the card from the packaging material look for the red LED (Light Emitting Diode) nearest one CORNER of the board. To orient the card correctly for installation, this LED must point toward the FRONT of the Peripheral Expansion Box. When the card is properly oriented, all of the IC's and other components will be facing the disk drive enclosure.

Install the card in any available slot -- just make sure that none of the components on either side of the card contact the "clam-shell" case of an adjacent card. This will avoid a short circuit.

4) LOADING THE RAMDISK OPERATING SYSTEM

Your System Master diskette contains all of the programs required to load the RAMDISK operating system. It is recommended that you make a back-up copy and keep the System Master diskette in a safe place.

The System Master diskette contains a program called UTIL1. This is an assembly language loader program for the RAMDISK operating system. It requires one of the following modules to execute: Extended Basic, Editor/Assembler or TI-Writer. If Extended Basic is used, the files LOAD and YLOAD are also required.

The operating system is contained in two files called PARTA_04 and PARTB_04. These files MUST BE PRESENT ON A DISKETTE WITH THE NAME ROS so the UTIL1 program can find them. (That is why the System Master diskette is named ROS -- ROS is an acronym for Ram Operating System. NEVER NAME THE RAMDISK ROS!)

The diskette, named ROS containing the above files may be placed in any drive EXCEPT DRIVE 3 (upon loading the operating system the RAMDISK becomes drive 3.) If using

Extended Basic place the disk in drive 1.

to execute the RAMDISK operating system loader:

A) From Extended Basic with the System Master disk in Drive 1.

1) Select Extended Basic from the powerup menu. LOAD will be executed automatically.

2) Select 1 from the System Master menu and press ENTER.

or B) From Editor/Assembler, with the System Master disk in any drive except drive 3.

1) Select E/A option 5 (RUN PROGRAM FILE)

2) Type: DSKn.UTIL1 press ENTER

or C) From TI-WRITER, with the System Master disk in any drive except drive 3.

1) Select TI-Writer option 3 (UTILITY)

2) Type: DSKn.UTIL1 press ENTER

UTIL1 may be used for single or multiple RAMDISK installations. Upon execution UTIL1 will display the CRU addresses from >1000 to >1700, indicate whether or not a Horizon Ramdisk is present at each address, indicate whether or not the RAMDISK operating system is loaded, and indicate the current setting for Drive Number, Maximum Sector and Number of Floppies for each Ramdisk in your PE-box. The arrow keys allow you to move the cursor around the screen. To load the operating system on a Ramdisk type a Y in the column for DSR at the appropriate CRU address. If for some reason you wish to disable the operating system type N in the field for DSR. This will have the effect of making that RAMDISK invisible to the computer. The default values for Drive Number, Maximum Sector and Number of Floppies when you load the operating system is drive 3, 720 sectors, and 2 floppies. These values may be altered by simply editing the screen. If you have a single-sided RAMDISK Maximum Sectors should be set for 360. Exit UTIL1 by pressing FUNCTN Q.

If you have loaded the operating system for the first time, you will need to initialize the RAMDISK before use. To do this, you can use most disk managers -- the Disk Manager II Command Module, for instance, works just fine if your RAMDISK is at CRU >1000. However, the RAMDISK WILL NOT WORK with disk managers designed to access ONLY the disk drive

controller card (such as the CorComp Disk Manager -- the CorComp Disk Manager Program will not recognize the HORIZON RAMDISK).

By special agreement with the Ottawa TI 99/4 Users Group, your System Master diskette also contains Version 3.3 of Disk Manager 1000 (DM-1000) written by Bruce Caron and revised by Ralph Romans. Documentation for DM-1000 is files DMDOCPT1 to 4 on the disk named SOURCE-B. The documentation files may be read with TI-Writer or printed. It is an excellent disk manager and, as will be explained more fully later, can be loaded directly from BASIC or Extended BASIC with your HORIZON RAMDISK.

The first time you load DM-1000, follow this procedure.

A) From Extended Basic with the System Master disk in drive 1...

1) Select Extended Basic from the powerup menu. LOAD will be executed automatically.

2) Select 2 from the System Master menu and press ENTER.

or B) From Editor/Assembler with the System Master disk in any drive except drive 3...

1) Select E/A option 5 (RUN PROGRAM FILE)

2) Type: DSKn.MGR1 press ENTER

or C) From TI-Writer with the System Master disk in any drive except drive 3...

1) Select TI-Writer option 3 (UTILITY).

2) Type: DSKn.MGR1 press ENTER

You should now see the DM-1000 master screen -- if you're at all familiar with the CorComp disk manager, you'll find DM-1000 very easy to use. Otherwise, PLEASE READ THE DOCUMENTATION FILES.

With DM-1000 loaded, you need to initialize your RAMDISK; keeping in mind the values set for drive number and Maximum Sectors when you loaded the operating system. If you have a single-sided RAMDISK, change the maximum number of sectors with CALL MS(360) from BASIC as described below before initializing.

After initializing your RAMDISK, use DM-1000's File Copy utility to copy two files from the System Master disk to the RAMDISK These two files, MGR1 and MGR2, contain the DM-1000 program. To leave DM-1000, use FCTN QUIT.

Once you have completed all of the above steps, you are ready to proceed with the remainder of the RAMDISK reference manual.

CALL SUBPROGRAMS AVAILABLE ON THE HORIZON RAMDISK

The RAMDISK contains nine CALL subprograms which you can use directly from TI BASIC or TI Extended BASIC. (Access from a running Extended program is effected using the special statement DELETE XBCALL as described below.) These subprograms can be used either within programs or in Command Mode.

1) CALL DN(n) - CHANGE RAMDISK DRIVE NUMBER

Whenever the RAMDISK operating system is installed, the RAMDISK will automatically respond as drive 3; i.e., to DSK3 access. However, with the CALL DN(n) statement you can change the number the RAMDISK responds to, and because it is battery-backed, it will respond to this number until changed with another CALL DN(n) statement. The "n" inside the parentheses MUST BE A SINGLE DIGIT FROM 1 TO 6. (Variable names are not allowed.) When the CALL DN(n) statement is executed, the RAMDISK will respond to any standard disk access to that drive number (DSKn.XYZ).

There are a few restrictions, however. First, no files can be OPEN in either the drive number currently assigned to the RAMDISK or the drive number to which the RAMDISK will be changed. For example:

A) If the RAMDISK were set to respond to DSK3;
B) You had a file OPEN on a diskette in DSK1; and
C) You then tried CALL DN(1) -- which would make the RAMDISK respond to any drive 1 access -- you would receive an INCORRECT STATEMENT error message because of the OPEN file on drive 1.

However, if you first CLOSE the file in drive 1, then CALL DN(1) will work (assuming no files are OPEN for drive 3, of course).

Remember, CALL subprograms can be used within a TI BASIC program; so if you encounter an INCORRECT STATEMENT error in a program which uses CALL DN(n) statements, check to make sure all files are CLOSED for both the drive number the RAMDISK was and the one it will be.

The second thing to be aware of involves what happens to "real" disk drive units normally assigned the number being used by the RAMDISK -- they cannot be accessed except by disk name. If, for example, you use a CALL DN(2) then all DSK2 access will reference the RAMDISK -- if you have a disk drive 2 attached to your system, it will become "invisible" until the RAMDISK drive number is changed. However, you can use the disk name option -- DSK.DISKNAME.FILENAME -- to access the disk drive which has the same number as the RAMDISK. (Another exception to the above is use of the CorComp Disk Manager and other programs that do not recognize the RAMDISK.)

2) CALL MS(n) - CHANGE MAXIMUM NUMBER OF SECTORS

This subprogram sets an upper limit on the number of sectors the operating system will recognize as valid. If you try to access a sector above this number (as in initialization of a disk) an error condition will occur. Once set, the number will remain in effect even after the system has been powered-down because the RAMDISK is battery-backed.

Your RAMDISK may vary with respect to the number of "sectors" of RAM actually on it. In general, if you have a "single-sided" RAMDISK, the maximum number of sectors is 360 and if it's double-sided the maximum is 720. NEVER SET THE NUMBER OF SECTORS HIGHER THAN THE ACTUAL AVAILABLE SPACE ON YOUR RAMDISK.

The CALL MS(n) statement allows you to inform the operating system as to how many sectors are available for use. Again, "n" must be a group of DIGITS -- not a variable name. For a single-sided RAMDISK, you would enter CALL MS(360) and for a double-sided one CALL MS(720).

When the operating system is loaded, it will automatically enter a default value of 720. If you have a double-sided RAMDISK, you can initialize it as double sided right away. Similarly, if you have a single-sided RAMDISK you may initialize it as single sided with no problem, but to avoid confusion it is a good idea to set the number of sectors at 360 before initializing.

3) CALL WO / WF - TURN DISK WRITE-PROTECT ON AND OFF

By entering CALL WO (write protect on) from BASIC you have, in effect, write protected the entire disk (this is much like putting a write-protect tab on a diskette). Similarly, by entering CALL WF from BASIC, you have removed the disk write protection. At the time the operating system is loaded, disk write-protection is "OFF".

Remember, the RAMDISK is battery-backed. If you write protect the RAMDISK today, it will still be write protected when you turn on your system tomorrow.

4) CALL EX(adr) - EXECUTE MACHINE CODE

This subprogram allows the user to transfer control of a BASIC or Ext. BASIC program to a specified address within the computer's CPU memory. The address must be DIGITS -- variable names are not allowed. All values are in decimal using the same conventions as TI's CALL LOAD subprogram. Since the power-up routine starts at >0024, it may be executed with the statement: CALL EX(36), for instance.

5) CALL CO / CF - TURN DSRam ON AND OFF

The CALL CO (card on) instruction enables the RAMDISK operating system. You will see the red LED light. At the same time it allows you direct access to 6K of the operating system software (from >4000 to >57FF) and the first eight sectors of the RAMDISK (>5800 to >5FFF). This information can be accessed directly from BASIC using the CALL LOAD and CALL PEEK subprograms in many modules. But even more importantly, in connection with the CALL EX subprogram (described as (4) above), you can directly link to operating system software from TI BASIC.

Before practical use can be made of this subprogram with respect to RAMDISK software, you will need the operating system source code and other information supplied with the RAMDISK. The CALL CO and CALL CF subprograms are described for those experimenters who will choose to do some work outside the RAMDISK environment as described in the "Advanced" section of this manual.

The CALL CF subprogram just shuts the card off, and of course the LED goes off.

6) CALL DM - LOAD DM-1000

This subprogram loads the DM-1000 disk manager from the RAMDISK if the RAMDISK contains the two files MGR1 and MGR2. If you have these two files on your RAMDISK, DM-1000 will load (rapidly, of course) and if you do not have the two files on your RAMDISK, the subprogram should act like a REM statement -- nothing will happen.

DM-1000 turns on drive 1 after it is loaded to look for a configuration file. This slows things down by a second or so. So if you want really fast access to DM-1000, make your RAMDISK DSK1 (using the CALL DN(n) subprogram described above.)

7) CALL NF(n) - SET NUMBER OF FLOPPY DRIVES

If the base CRU address is set at the normal value of >1000, then the CALL NF statement is of no use.

When the RAMDISK base CRU address is not >1000, a special quantity must be specified if DM-1000 is to access the RAMDISK. This value is set equal to the number of actual floppy drives attached to the user's system. If the RAMDISK drive number is set greater than this value, then DM-1000 will be able to access it.

If the CRU base is not >1000, then BASIC and other programs accessing disk drives can access the RAMDISK only when its number is greater than the MAXIMUM number of drives the disk drive controller can access (3 for TI controller and 4 for CorComp). However, by proper specification of NF, DM-1000 can access the RAMDISK at any number from 2 to 6.

When the Operating System is loaded, NF has the value of 2. In the CALL NF(n) statement, (n) is a digit between 1 and 5. Variable names are not allowed.

8) DELETE "XBCALL" - DOWNLOAD MACHINE LANGUAGE FOR XB LINKS

The CALL statements as described above will not execute from a running Extended BASIC program. Program mode in Ext. BASIC does not scan PE Box cards for CALL routines. The DELETE "XBCALL" statement, when executed immediately after a CALL INIT statement, downloads the machine language for all CALLs into the low 8K of the 32K memory expansion unit.

The routines are then accessed using CALL LINK statements of analogous form to the CALL statements above:

TI BASIC:		Extended BASIC:
CALL DN(n)	-->	CALL LINK("DN",n)
CALL MS(n)	-->	CALL LINK("MS",n)
CALL DM	-->	CALL LINK("DM")
	etc.	

The DELETE statement will execute from a running Ext. BASIC program: RAMDISK CALL LINK statements can be used in auto-load and chained programs.

The DELETE "XBCALL" statement MUST immediately follow a CALL INIT statement. If a CALL LOAD("DSKN.PROGNAME") statement is executed in between the CALL INIT and DELETE "XBCALL", no machine language is downloaded (and no error message is returned). However, other assembly language programs can be CALL LOADed after the DELETE if they are re-locatable code.

9) CALL ? - YOUR CALL ROUTINE(S)

The source code for the Operating System, together with the information in the "Advanced" section of this manual will allow assembly language programmers to add more CALL routines to those above.

Those who care to experiment with CALL routines and have sufficient knowledge of assembly language are invited to do so. This can be done either within or outside the RAMDISK operating environment.

COMMON RAMDISK APPLICATIONS

In general, the HORIZON RAMDISK functions just like a standard floppy disk drive -- only faster. Here are a few common application areas in which the RAMDISK saves considerable time and makes the TI 99/4A more fun to use.

1) TI WRITER

Anyone who uses TI Writer will appreciate the increase in speed and productivity which can be attained using the HORIZON RAMDISK. The base CRU address must be set at >1000 (see Installation Instructions for an explanation of how to set the CRU base address).

After installing the operating system and initializing the RAMDISK, copy the following TI / TK Writer files onto your RAMDISK:

A) EDITA1

B) EDITA2

C) FORMA1

D) FORMA2

E) CHARA1 (if using TK Writer)

Now make your RAMDISK respond to drive 1 by entering the BASIC command CALL DN(1). Exit BASIC, insert your TI Writer Module, and you're ready to begin.

You will notice that the Editor and Formatter now load in about one second. There will still be plenty of sectors left on the RAMDISK for saving your document while you are working on it -- and you should notice a definate increase in the speed with which you are able to save and load your text files using the RAMDISK.

2) MULTIPLAN

As with TI Writer, the CRU base address should be >1000 (see Installation Instructions for further information). Copy the Multiplan System Disk that came with the TI Multiplan Module onto the RAMDISK. Make sure the disk name is TIMP as is required by the Multiplan system. Make your RAMDISK drive 1 using the CALL DN(1) statement. Now all program accesses by Multiplan will be directed to the RAMDISK greatly decreasing the time required.

Your spreadsheets can, of course, be saved to the RAMDISK if you wish. But the real advantage is in the time saved in faster accesses to the OVERLAY and HELP files.

3) TI BASIC AND EXTENDED BASIC

The HORIZON RAMDISK is fully compatible with TI BASIC and Extended BASIC. You can load and save programs, open, close, read from, and write to all types of files. The CRU base address need not be set at >1000 for use in the BASIC and Ext. BASIC environments; however, a higher address (e.g., >1200) would allow use of only DSK4, 5, and 6 -- and only DSK5 and 6 for those with the CorComp Disk Controller.

With BASIC and Ext. BASIC the speed advantage of the RAMDISK is not as apparent because although disk access may be happening 20 times faster than normal, things like prescan, the BASIC and GPL interpreters, etc. require significant amounts of time. These factors are, of course, not affected by the presence of the RAMDISK.

The RUN "programfile" Extended BASIC statement works fine with the RAMDISK. If you write your programs to minimize prescan time as described in the Extended BASIC manual, the RAMDISK can provide some real speed advantages when chaining program segments. Remember that in Extended BASIC programs, use of the RAMDISK CALL routines requires the DELETE "XBCALL" statement as described in the CALL instructions.

As with standard disk drives, files can be accessed by disk name using the RAMDISK. You can use a statement like:

```
200 OPEN #2:"DSK.RAMDISK.FILENAME"
```

Files can be cataloged from BASIC using the RAMDISK using the same procedure as with a standard disk drive (see pg. 37 of the TI Disk Memory System manual.) This allows use of all standard menu/auto-load programs.

4) EDITOR/ASSEMBLER

After initializing your RAMDISK, copy the following files onto it:

A) EDIT1

B) ASSM1

C) ASSM2

Use the CALL DN(1) statement to make the RAMDISK respond to DSK1 access. You can load the Editor or the Assembler in about one second! You can also save your source code on the RAMDISK and assemble the object code directly onto it. This results in assembly time about one-third that of a floppy drive. The RAMDISK supports all Load and Run and Run Program File functions of the Editor/Assembler and Mini Memory Command Modules.

IN CASE OF DIFFICULTY

Problems and "bugs" of various kinds have been found with many all computer products, and the HORIZON RAMDISK will probably be no different in this regard. However, because the operating system for the card is in RAM, we can correct software problems by sending you a new disk. You can help us to improve your RAMDISK by letting us know about the kinds of problems you experience in a way that will help us identify and correct the problem.

Whenever you experience a problem in using your RAMDISK, there are several questions you should try to answer. The first is:

Is the Problem the Repeatable

Using the same piece of software or the same disk, can you make the problem happen consistently. (While ramdom problems are difficult to diagnose, we would still like to know about them.) If the problem is repeatable, try making it happen after powering-down your system and waiting several minutes. Also try it immediately after re-loading the operating system (this will not affect the contents of your disk). Any information you can provide to pin down the factors which cause the problem will be of help.

Is the Problem Specific to the RAMDISK

If you make a sector copy of the RAMDISK contents to a floppy, set the RAMDISK number at 6, and try the problem situation again using a floppy drive in place of the RAMDISK, does the problem occur? If so, it may not be related to the RAMDISK. In the process of RAMDISK software development there have been many times we thought there were problems with the RAMDISK only to discover the true source of the problem was elsewhere in the system.

One case in point involved intermittent loss of the operating system or parts of it. This was traced to a poor connection between the speech synthesizer and the flex cable. Removing the speech synthesizer solved the problem.

Is the Problem Hardware Related

Memory chips can fail, and so can any of the other components of your RAMDISK. In order to facilitate a test of your RAMDISK memory, a TI BASIC program called MEMTEST has been included on the System Master diskette. MEMTEST wipes

out the ENTIRE contents of the RAMDISK including the operating system, so make sure you copy important files to a floppy diskette before running MEMTEST. After running MEMTEST, the operating system must be reloaded.

To execute MEMTEST:

A) From Extended Basic with the System Master disk in drive 1...

1) Select Extended Basic from the powerup menu. LOAD will execute automatically.

2) Select 3 from System Master menu and press ENTER.

or B) From BASIC with the Editor/Assembler or Mini-memory command module inserted and the System Master disk online.

Type: >OLD DSKn.MEMTEST
>RUN

MEMTEST first asks you to select a CRU address corresponding to the DIP switch setting of the RAMDISK you wish to test. The program then asks whether you want to test the U11 chip or the entire memory for a single or double sided RAMDISK. It then proceeds to test the memory by writing and reading a series of eighteen special numbers to each of the 8K chips on your card to see if they retain the values written. If you have a single-sided RAMDISK, 13 chips will be tested for each test number -- 24 for a double-sided.

You may find one or more bad memory chips. Alternatively, you may find that for certain numbers several chips appear bad. An even more rigorous test is the Disk Manager II comprehensive test. Test 6 of the series has been able to locate problems on double-sided cards that no other test detects. All such problems have been due to poor solder connections on piggy-backed IC's. If you find errors, contact us regarding repair of your card.

If you do not find errors, the problem is likely to be in the software which controls the RAMDISK. When you have verified that the problem is repeatable, that it is specific to the RAMDISK, and that your hardware tests OK, please notify us of your problem so that it can be corrected.

No Access to Floppy Drives

If your system appears normal on power-up, but locks up when you try to access your floppy drives, re-load the operating

system as follows: Power-down and wait two minutes. Remove the card and turn DIP switch 1 to the OFF or OPEN position. Turn switch 3 to the ON or CLOSED position to set the CRU base address at >1200. This will allow the disk controller card to be accessed before the RAMDISK. Re-install the card and load the operating system. Power-down again and wait two minutes. Then remove the card and re-set the DIP switches as desired.

System Lock-Up on Power-Up

Occasionally (especially if you choose to experiment with writing your own routines for the card) you may find that when you turn on your computer, you get a blank screen and that the computer is "locked-up". If you find that this happens only when the RAMDISK is plugged into the PE-Box, bad data has found its way into the RAMDISK operating system memory, and the operating system must be re-loaded. (Power down and WAIT TWO MINUTES before removing the RAMDISK card.)

Before the operating system can be re-loaded, the card must be made "invisible" to the system. If you have Mini Memory or DEBUG on a SUPER-CART module, simply turn on the card (using the CRU command and entering a 1 for the appropriate base address). When you see the LED light, set the contents of CPU memory address >4000 to 0 (it should be >AA01 initially).

Alternatively, you can follow this procedure in BASIC (with Editor/Assembler or Mini-Mem) or in Extended BASIC (with 32K memory expansion.)

1) Power-down the console and PE-Box;

2) Turn on the console FIRST, then the PE-Box;

3) Enter TI BASIC (or Extended BASIC); and

4) Enter and RUN this program:

```
100 CALL INIT

110 CALL LOAD(12288,2,12,16,0,29,0,4,224,
    64,0,30,0,4,224,131,196,2,224,131,224,
    4,32,0,0,"",-31804,48,0)
```

5) This program when RUN should return to the Master Title Screen. If it works, power-up the system in the normal fashion -- PE-Box FIRST, and you should get a normal screen.

6) Re-load the RAMDISK operating system.

If this program fails to work the only other alternative is

to power-down the system, remove the RAMDISK, and take out one of the three NI-CAD batteries. Allow the card to stand for at least 15 to 30 minutes so the memory contents are lost. Then re-insert the battery and place the card back in the PE-Box. (RUNNING THE CARD WITHOUT ALL THREE NI-CAD BATTERIES IN PLACE MAY CAUSE DAMAGE TO THE MEMORY CHIPS!)

Your system should now power-up normally. Re-load the operating system.

Should you need further assistance, you may write to the company or call one of the individuals below (sorry, but our extremely low margins will not allow us to accept collect calls.)

Ron Gries (419) 874-1414
David Romer (419) 666-6911

ADVANCED APPLICATIONS

The following portion of the manual is provided for those hobbyists, experimenters, and hackers (and software developers) who want to do some programming on their own outside the environment of the RAMDISK application software. Addition of CALL routines to the existing operating system can be done by referring to the source code provided. However, a detailed discussion of how to do this is beyond the scope of this manual.

While you can use the information provided to write your own CALL statemements for BASIC and Extended BASIC (in command mode only) or experiment with power-up, interrupt, or device service routines, you will first have to erase the program which makes the card a RAMDISK -- the ROS. What the card does then depends upon what software you write for it. (The ROS can be reloaded at any time as described under Installation.)

INTRODUCTION

The Ram Disk operates entirely within the address space reserved for Device Service Routines (DSR's), viz. >4000 to >5FFF. The card may be thought of as being in one of two states -- ON or OFF. Any time the card is on, the LED will be lit on the front of the Peripheral Expansion Box, and if the LED is not lit, the card is OFF.

The card is enabled or disabled (turned on or off) using the Communications Register Unit or CRU. Understanding exactly how the CRU works can be very difficult. Luckily, using the CRU turns out to be very simple and requires little if any knowledge of how it works.

As an assembly language programer, you may have noticed the LDCR instruction, and may have even had occasion to use it. The LDCR instruction is all that is required to access all memory segments of the HORIZON RAMDISK card.

ARCHITECTURE OF THE CARD

The 6K DSR Section

Figure 1 depicts the RAMDISK memory map. A 6K segment >4000 to >57FF, is enabled every time the card is enabled. This 6K block holds most of the ROS. There is only one memory block that can be enabled in this address space.

The 2K RACK

The remaining 2K in the DSR space is from >5800 to >5FFF. In the >4000 to >57FF space, there is only one block of memory that can be turned on for read-write operations.

By contrast in the remaining 2K area, >5800 to >5FFF, any one of many blocks may be switched on for reading or writing data. The 2K sections that can be turned on and off in this address space are called RACKS (RAm traCKS). RACKS are numbered. The first RACK is 0 and the last one is 92.

The RAMDISK Sector

The RAMDISK application further divides each physical RACK into logical sectors -- there are eight 256 byte sectors per RACK (See Figure 2). This division of the RACK is purely arbitrary in the sense that it is an artifact of our software. A non-ramdisk application would, no doubt, use a different logical organization for the RACKS.

Single Sided vs. Double Sided

If you have a single sided RAMDISK, you have all the blocks in Figure 1 which have the heavy black line beside them. This includes the first 6K, RACKS 0 through 44 and RACKS 89 through 92. Double siding a card also adds RACKS 45 through 88.

Notice that even if you have a single sided card, you have RAM for a few sectors of the end of a double sided configuration -- 712 to 719. In addition, there are three RACKS which are used as integral parts of the ROS.

The Last Three RACKS

RACK 90 contains portions of the ROS. Although we started out believing we could fit the program to operate the RAMDISK in 6K, we needed nearly 8K. Portions of the code which do not require disk sector access have been moved to RACK 90.

RACK 91 contains all of the CALL subprograms you can access from BASIC and Extended BASIC. The only exception is CALL DM. There is plenty of room left in this block for more CALLs or data.

RACK 92 contains the character patterns used by DM-1000, starting with the cursor, ASCII 30. Following this data is the machine language that gets downloaded for DELETE "XBCALL". Finally, RACK 92 contains the program that transfers this machine language into the low 8K section of the memory expansion unit. There is additional room in this RACK for more CALLs or data.

HOW TO ENABLE THE 6K AREA AND A RACK

Loading the CRU Base Address

Whenever a CRU access is made, the CRU base address must be loaded in Workspace Register 12. If the DIP switch settings on your card are standard, the CRU base address of the card is >1000 -- you would place >1000 in R12. If your card is set for CRU base address >1200, you would place that value in R12.

Calculating the RACK Code

Assuming the desired RACK number is known (0 - 92), the CRU address of the RACK is found as shown below Figure 2 -- 2 * RACK + 7. In a RAMDISK example, the stating quantity would be the sector number (0 - 359 or 719). The RACK containing that sector is simply the integer value obtained by dividing the sector number by 8.

Enabling the DSR and RACK

Once the RACK code is calculated (2 * RACK + 7), it is placed in the most significant byte of a word, and the LDCR @WORD,8 instruction is executed. This will enable the 6K DSR section and the RACK selected.

Suppose one wanted to enable RACK 92 (and the 6K DSR) in an assembly language program:

```
BASE   DATA >1000       ;Base CRU address set on DIP switch
*
* First Load CRU Base Address
*
START  MOV  @BASE,R12   ;Put base CRU accress in R12
*
* RACK CRU = 2 * RACK + 7
*
       LI   R1,92       ;Put desired RACK in a register
       SLA  R1,1        ;Multiply by 2
       AI   R1,7        ;Add 7
*
* Put Code in Most Sig Byte of a Word
*
       SWPB R1          ;swap most and least sig bytes
*
* Enable 6K DSR and RACK 92
*
       LDCR R1,8
*
* Now >4000 - >57FF is the DSR RAM
*
* and >5800 - >5FFF is the 2K in RACK 92
*
```

Now the address space >4000 to >5FFF can be accessed as ordinary CPU RAM -- it can be used to load and run machine language, store and fetch data, etc. When the DSR is disabled, and the computer system powered down, however, everything written to this area will be retained under battery back-up.

Disabling the Card

To disable the card, load R12 with the base CRU address (usually >1000) and then simply use a SBZ 0 instruction.

```
*
* Load R12 with CRU Base Address
*
      LI   R12,>1000
*
* Disable the Card
*
      SBZ  0 (PS - an SBO here will turn ON
                   the card and RACK 0)
```

If you have accessed the card via a standard DSRLNK routine, that routine will turn off the card for you upon return. However, if you have turned it on explicitly, then you must turn it off via your program as described above. Remember that the card must always be turned off from a program running in memory space outside >4000 to >5FFF -- if you try to have the card turn itself off the computer will be faced with a bunch of 0's for machine language instructions and quickly run amok.

LOADING YOUR OWN PROGRAMS

On your System Master diskette, you will find a few programs that may prove useful in experiments with the DSR area. These include a program to erase the 6K DSR and RACKS 90-92 (leaving unaffected the actual contents of the RAMDISK), and the source and object code for a special loader which will allow you to load object files created with Editor/Assembler directly into the 6K and three 2K blocks.

CLEARING THE OPERATING SYSTEM

You may disable the RAMDISK operating system by executing the UTIL1 loader program and typing an N in the column that indicates the status of the operating system. The >AA01 in the DSR header is removed making the RAMDISK invisible to your computer. Any information actually contained in the SECTORS of the RAMDISK will be un-affected by this operation so that immediate restoration of the operating system via UTIL1 would again allow access to all disk information. (Naturally, if programs you write for the DSR modify sector data, this will not necessarily be the case!)

The LOADER

Located on your System Master disk is a source code file called LOADER/S. When assembled and run this program loads tagged object files created with Editor/Assembler. Several restrictions are involved, however. First, you MUST use only Absolute Origin (AORG) code. You can use as many AORG statements in your program as you like and can even place code outside the >4000 to >5FFF DSR space, but LOADER will not handle relative code.

Secondly, you must not use compressed object files. The loader only works with uncompressed object files. Do not specify a "C" in the options prompt when assembling your program.

Finally, on a given execution, this LOADER makes available the standard 6K (>4000 to >57FF) and one of the remaining 2K blocks. You are asked to select which of the 2K blocks you want enabled when the loader is run. Each of the 2K blocks (RACKS 90-92) are decoded from >5800 to >5FFF. If code is to be placed in more than one of the three 2K blocks, this must be done via separate executions of the LOADER program and separately assembled object files. Code destined for a 2K block must be AORG'd with a value of >5800 to >5FFF (just as code for the 6K area must be AORG'd from >4000 to >57FF.)

You first write and assemble your DSR program. Save the un-compressed object file on diskette (NOT RAMDISK). Then, with the LOADER program on-line, select Option 3 (Load and Run) from the E/A Main Title Screen and enter the File Name: DSKx.LOADER

You will first be asked to enter the name of the object file you want loaded. First key in the disk drive number, then a decimal point, and finally the object file name used when the program was assembled. You can use FCTN arrow keys to move the cursor forward and backward. When the name is correct, press ENTER.

Then, if the file is found, you will be asked to select which of the three 2K blocks AORG'd code from >5800 to >5FFF (if present in your program) is to be placed. RACKS 90 to 92 are 2K blocks 1 to 3, respectively. Press key 1, 2, or 3, depending on where you want the code to go. If you have no code AORG'd over the >5800 area, it does not matter which 2K block you select.

After the file is loaded, the loader provides the highest address used in both the 6K and 2K block. From this information you can insure that you have not exceeded the limits of the block and determine just how much space is left in that block. If the highese address used for a block is

>0000, no code was loaded into that block. When using the LOADER, you need not load code into either the 6K or 2K block -- what gets loaded will depend upon your AORG statements.

After the code is loaded, you will be asked to press ENTER to continue. Press ENTER, then FCTN QUIT or FCTN BACK.

EXPERIMENTING WITH DSRs

A full discussion of DSR block structure is beyond the scope of this manual. A supplementary manual is planned. However, full information can be found in TI's manual: TI-99/4A CONSOLE AND PERIPHERAL EXPANSION SYSTEM TECHNICAL DATA. Pages 54 and 55 give Assembly Language examples of DSR header formats and the accompanying text describes the techniques used to enter the DSRs as well as for returns to calling programs.

One area NOT covered in the TI manual, however, is the writing of routines which can be placed on the card and accessed via CALL statements from TI BASIC or TI Extended BASIC. The remainder of this section of the manual will describe the techniques involved.

THE CALL STATEMENT

Whenever the BASIC interpreter executes a CALL statement, it searches through all potential CRU addresses to see if there are any valid DSR cards at those addresses. When it finds a card, it gets the address of the CALL Table. The CALL Table contains a list of CALL names (like SOUND, COLOR, SPRITE, etc.) Associated with each name is the address at which that routine starts and a pointer to the next name in that list (if any).

After searching all cards, the computer searches GROMS to find the name. If at any point in the search the name is found, that routine is executed and control is returned to the BASIC interpreter.

In order to write assembly code which will be loaded onto the HORIZON card so as to be executed via BASIC CALL statements, use the following following set-up program:

```
*****************************************************************
*
* The example program below shows the general techniques used in
* writing CALL routines.  These two are CALL A and CALL BBB which,
* respectively, fill the screen with A's and B's when executed from
* BASIC or Ext BASIC.
*
* To try them, first run the CLEAR program.  Then used LOADER to load
* the object file (CALLDEMO).  (It doesn't matter which 2K block you
* select.)  Once loaded you can try them out from BASIC.
*
*---------------------------------------------------------------
*
* General Header For DSRs
*
*---------------------------------------------------------------
*
        AORG >4000          Start of 6K area
        BYTE >AA            Code for valid DSR
        BYTE >01            Version number = 1
        DATA 0              not used
        DATA 0              POWER-UP LINK table address
        DATA 0              not used
        DATA 0              DSRLNK - used for OPEN, CLOSE etc
        DATA CTABLE         CALL ROUTINE LINK table address
        DATA 0              INTERRUPT ROUTINE LINK
        DATA 0              not used
*
* ---------------------------------------------------------------
*
* The Link Table for CALL Routines
*
CTABLE  DATA NEXT1          LINK to next entry in table
        DATA FILLA          ENTRY point for first routine
        BYTE >01            Name length - 1 char in name
        BYTE 'A'            Name of routine - CALL A
NEXT1   DATA 0              A LINK of 0 means this is the last entry
        DATA FILLB          ENTRY point for second routine
        BYTE >03            Name length - 3 bytes
        TEXT 'BBB'          CALL BBB is name of this routine
        DATA 0
*
*---------------------------------------------------------------
*
* Use these three DATA statements for all CALLS
*
MASK    DATA >4000          Mask for floating point error code
SAVADR  DATA 0              Holds return address
TOKEN   DATA 0
*
*---------------------------------------------------------------
*
* Note that both routines below start out by saving the return addres
* and resetting the token pointer to the end of the BASIC line.
*
*---------------------------------------------------------------
```

```
*-----------------------------------------------------------------------
*
* START OR FILLA
*
FILLA  MOV   R11,@SAVADR   Save return address
       MOV   @>832C,R1     >832C -> addr of BASIC token code pointer
*                          this points to the byte immediately AFTER
*                          the CALL token in the BASIC program.  By
*                          advancing this value you may read data
*                          directly from the remainder of the program
*                          statement.  Ultimately, it must point to the
*                          End-of-Line mark "0" in the BASIC program an
*                          in this case the name only contains 1 char.
       INC   R1
       MOV   R1,@TOKEN     Save in TOKEN for return
*
* Now do the routine - fill the screen with A's
*
       LI    R1,>A100      "A" 65+96 offset for basic
       CLR   R0            Starting VDP RAM address
       SOC   @MASK,R0      indicate a write
       SWPB  R0            LSB First
       MOVB  R0,@>8C02     Move into VDPWA
       SWPB  R0            Now MSB
       MOVB  R0,@>8C02     to VDPWA
       LI    R0,768        768 "A"s to write
ALOOP  MOVB  R1,@>8C00     Write a byte to VDPWD (auto incrementing)
       DEC   R0            Done?
       JNE   ALOOP
       B     @RETURN       Done, use common return routine
*
* START OF FILLB
*
FILLB  MOV   R11,@SAVADR   Save return address
       MOV   @>832C,R1     Get token pointer
       AI    R1,3          This name is 3 chars so must add 3 to
*                          get to the end of the BASIC line
       MOV   R1,@TOKEN     Save token pointer
*
* Now do routine
*
       LI    R1,>A200      "B" 66+96 offset for basic
       CLR   R0            Starting VDP RAM address
       SOC   @MASK,R0      indicate a write
       SWPB  R0            LSB First
       MOVB  R0,@>8C02     Move into VDPWA
       SWPB  R0            Now MSB
       MOVB  R0,@>8C02     to VDPWA
       LI    R0,768        768 "B"s to write
BLOOP  MOVB  R1,@>8C00     Write a byte to VDPWD (auto incrementing)
       DEC   R0            Done?
       JNE   BLOOP
       B     @RETURN       Done, use common return routine
*
*-----------------------------------------------------------------------
```

```
*-----------------------------------------------------------------
*
* You can use the following code in any program.  If you
* need to return with an error, B @ ERTN.  Otherwise,
* B @RETURN.
*
* Error Return - use for INCORRECT STATEMENT error
*
ERTN    LI    R0,>0700
        MOVB  R0,@>8342     Make >07 the current token/char
        JMP   RTN
*
* Non-error Return
*
RETURN  SZCB  @>8342,@>8342 Make >00 (End of Line) the current token
RTN     MOV   @SAVADR,R11   Restore Return Address
        MOV   @TOKEN,R1     Get Token Pointer
        SWPB  R1            LSB First
        MOVB  R1,@>8C02     Move to VDP Write Address
        SWPB  R1
        MOVB  R1,@>8C02     Now do MSB
        MOV   R1,@>832C     This resets token pointer
        INCT  R11
        SZCB  @MASK,@>8354  Floating point error
        B     *R11
        END
```

MEMORY MAP OF HORIZON RAMDISK OPERATING SYSTEM

VERSION 04

>4000 - >57C7 Main DSR

1st 2K Block: RACK 90 - LINK1 = >BB00

>5800 - >5EDB Portions of DSR code that do not require access to RAMDISK contents. These are accessed via PTCHxy labels and return via RTNX labels.

2nd 2K Block: RACK 91 - LINK2 = >BD00

>5800 - >5B01 CALL Subprograms except CALL DM which is in PARTD.

3rd 2K Block: RACK 92 - LINK3 = >BF00

>5800 - >5B0F Pattern definitions for 98 characters used by DM-1000.

>5B10 - >5E19 Machine language for the Extended BASIC analogs of the CALL subprograms.

>5E1A - >5E61 Definition table showing the names of all the XB CALL analogs.

>5E62 - >5ED1 DELETE "XBCALL" code used to download the XB call analogs.

- DISK CONTENTS -

PARTA-E - Source code for five parts of the main DSR. Use the COPY directive in E/A to assemble into one object file.

CALL/S - Source for CALL subprograms (except CALL DM).

CHAR/S - Pattern definitions used by CALL DM.

XB/S - Extended BASIC analogs of CALL subprograms - low 8K.

DOWNLD/S - DELETE "XBCALL" routine.

SVXB/S - Uploads the XB/O code to its place in the third 2K block. Program name is SVXB. Executed as follows:

```
>CALL INIT
>CALL LOAD("DSK1.XB/O")
>CALL LOAD("DSK1.SVXB/O")
>CALL LINK("SVXB")
```

CREATE/S - Creates PARTA_xy and PARTB_xy memory image files holding whatever is in the 12K of DSR space. (>4000->5FFF plus the three 2K blocks).

LOADER/S - Source code for loader for uncompressed tagged object code AORG files.

PARTA_04
PARTB_04 - Memory image files of DSR. Used by UTIL1.

MGR1
MGR2 - Memory image files for DM-1000 Ver. 3.3.

LOAD - Extended BASIC auto-load program for UTIL1, DM-1000 and MEMTEST. Requires YLOAD file.

UTIL1 - Loader for Ramdisk Operating System.

MEMTEST - Test program for Horizon Ramdisk. TEST/O file.

DOCUMENT - Details DSR subroutines as to Work Space, Register usage, symbolic addresses, etc.

DMDOCPT1
DMDOCPT4 - Documentation for DM-1000.

Figure 1

RAM DISK MEMORY MAP

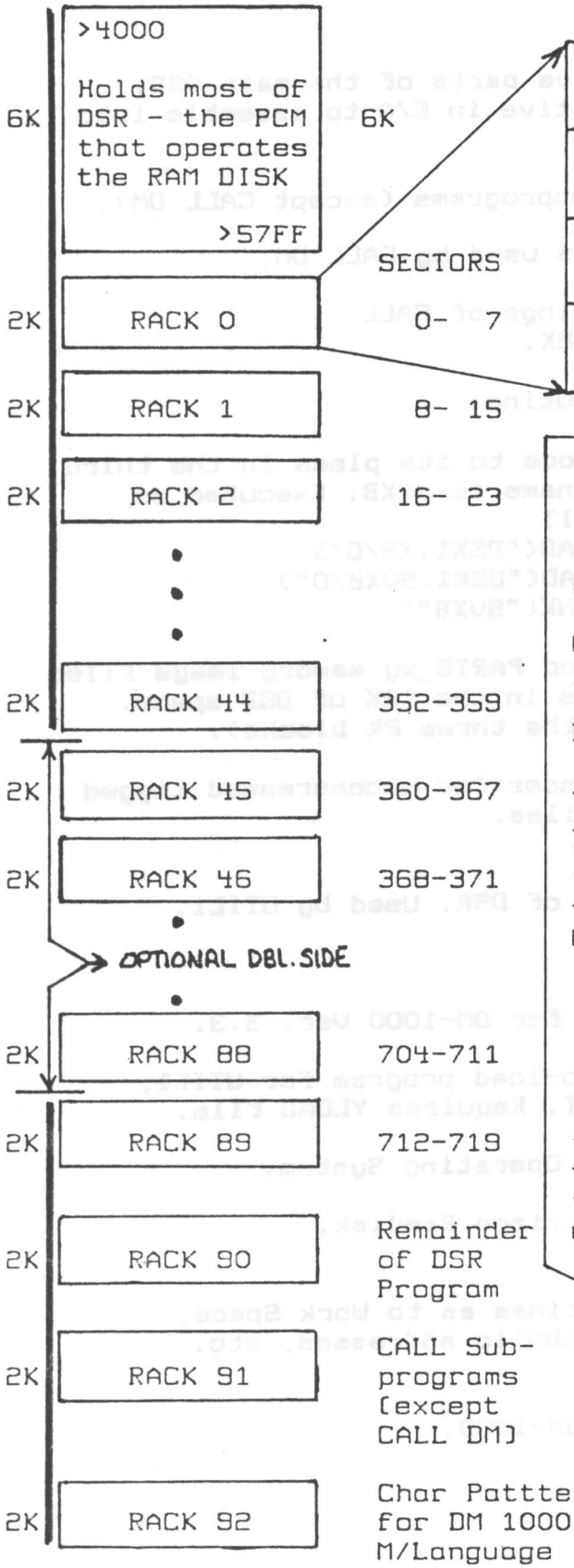
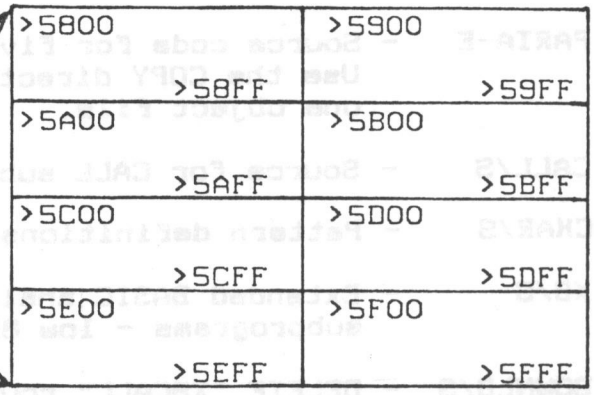


Figure 2

SECTOR STRUCTURE of RACK




```
IF SEC = Sector Number (0 - 719),
        RACK = INT(SEC/8)

CRU Addr RACK = 2 * RACK + 7

CPU Addr SEC  = >5800 + 256 * SEC MOD 8
-----------------------------------------
Routine to 1) Input SECtor number in R6
           2) Enable RACK
           3) Return CPU addr. of SEC in R6
-----------------------------------------
*
* CALLING PROGRAM
*
CRU     DATA >1000   :CRU Base Addr
        LI   R6.320  :SECtor 320
        BL   @ADDR   :Do 1) - 3) above
        MOV  *R6,R1  :Get first word 320
        .
        .
        .
*
* Routine to Enable RACK, Point to SECtor
*
ADDR    MOV  R6,R7     :Copy to R7
        SRL  R7,3      :INT(SEC/8)=RACK
        SLA  R7,3      :8*RACK
        S    R7,R6     :SEC MOD 8 (rmder R6)
        SLA  R6,8      :256 * SEC MOD 8
        AI   R6,>5800:+Base CPU addr in R12
        SWPB R7        :Put in MSB of word
        MOV  @CRU,R12:CRU Base addr in R12
        LDCR R7,8      :Enable RACK
        RT             :R6 has CPU of SEC
```