Programming EPROMs for GRiDCase and Compass

October 1986

2535 Garcia Avenue
P.O. Box 7535
Mountain View, CA 94039-7535
(415)-961-4800

Manual Name: Programming EPROMs for GRiDCase and Compass
Order Number: 031031-44
Issue Date: October 1986

Contents

List of Figures and Tables

About This Manual

The ROM (Read-Only Memory) capabilities of the GRiD Compass and GRiDCase provide a compact and durable storage medium for portable applications. This manual describes those capabilities, and the process by which an application is put into ROM.

GRiD supplies some of its operating systems and applications in executable ROMs. Software in an executable ROM saves RAM in the computer by executing directly from the ROM rather than being drawn into, and executed from, RAM. The process of making an executable ROM is quite difficult and only available under GRiD-OS. Therefore, it is not described in this document. If you desire to put your application into an executable ROM, contact your GRiD sales representative for consulting services.

Who Should Use This Manual

This guide is intended for the computer literate GRiDCase or Compass user with a desire to put an application into ROM. Software developers should consult this guide before developing a new application for ROM. Software integrators should also consult this guide if they want to move a pre-existing application from diskette (or other media) into ROM.

Prerequisites

It is assumed that the reader of this manual is familiar with software concepts on computers. A GRiDCase or Compass with 512 KB of memory and a hard disk are required to run the software necessary to put an application into ROM. Compass users with a serial PROM programmer may need a cable to convert the DE-19 (19 pin) serial connection on the Compass to a DB-25 serial connection. GRiD supplies a cable for this purpose; the model number is 6100.

How to Use This Manual

This manual is not intended to be read from cover to cover. Readers should peruse only those sections that apply to their computer, operating system and application.

Chapter 1 provides an introduction and overview to ROM burning.

Chapter 2 provides an overview of GRiDCase ROM capabilities.

Chapter 3 provides an overview of Compass ROM capabilities.

Chapter 4 provides information on preparing for and creating MS-DOS ROMs. The MS-DOS ROM software is described in detail.

Chapter 5 provides information on creating GRiD-OS ROMs. The files provided with the GRiD-OS ROM burning software are explained in detail.

Appendix A provides helpful information on handling EPROMs.

Appendix B provides GRiD part numbers for EPROMs and related items.

Appendix C provides specific information about particular PROM Programmers.

Appendix D provides additional technical information on creating EPROMs.

Conventions Used in This Manual

In process diagrams such as Figure 5-1, rectangular boxes are used to represent data files, and parallelograms (slanted rectangles) are used to represent programs or run files.

Other Documentation

Additional information on creating ROMs may be obtained from the following documents.

GRiDCase Technical Reference Manual	029500-50
GRiD-OS & MS-DOS Utilities	021210-43
GRiDTerm and GRiDReformat	021141-43
Using MS-DOS on the GRiD Compass	021040-43
Using MS-DOS on the GRiDCase (version 2.11)	029500-44
Using MS-DOS on the GRiDCase (version 3.2)	029550-44

Chapter 1: Introduction/Overview

The GRiD computer line of products is intended for use in very portable applications. The Compass and GRiDCase are compact, lightweight computers that carry a limited amount of internal storage. The Compass has 384 KB of bubble memory, and the GRiDCase has a 720 KB 3-1/2" floppy drive. In many applications this storage is adequate for data, and the application software is stored in Read Only Memory (ROM). The ROMs take the place of a second drive, being both lighter and more rugged than a floppy diskette. One of the major advantages of ROMs is that end users cannot accidentally damage the application software. It is also much easier to train end users to work with an application in ROM (they don't have to worry about inserting the application diskette). For these reasons, GRiD has designed the ROM capability for their machines.

ROMs are integrated circuits that contain code and data in much the same way as regular system memory (RAM). One difference is that ROMs are read-only -- they cannot be written to while installed in a Compass or GRiDCase. The second main difference is that ROMs are permanent storage, where RAM is transitory (its contents are lost when the power goes off). ROMs are lightweight and compact, as well as reasonably priced. They are intended for use with fixed applications in the GRiD line of computers.

GRiD has put its operating systems and many of its own utilities and applications into ROM. As with all computer systems, a GRiD system always requires an operating system to function, so an operating system ROM is a good example of effective use of the ROM capability. In most portable business systems there is a fixed application (account auditing, sales reporting, data collection, etc.). The users rarely require momentary use of different applications. These fixed applications are a perfect fit for ROM use.

The other advantages to ROMs are that they are durable, fast and prevent a novice user from accidentally erasing an application, since ROMs cannot be erased once they are installed in the computer.

In this manual the term ROM (Read Only Memory) is used interchangeably to mean EPROM (Erasable Programmable Read Only Memory) and/or PROM (Programmable Read Only Memory). Where a precise definition is needed, the exact term (EPROM or PROM) is used.

The different types of ROMs may be mixed together (PROMs and EPROMs) as well as different sizes (32 KB, 64 KB and 128 KB), different software types (bootable, executable and draggable), and different operating systems (MS-DOS and GRiD-OS). All of these terms are explained in later chapters, as well as in the glossary in the back of this manual.

Chapter 2: GRiDCase ROM Capability

The GRiDCase computer family has four 28-pin ROM sockets accessible under a small panel located above the keyboard. These sockets are referred to as the external ROM sockets. It also has four 28-pin ROM sockets on a ROM board "under the hood" (inside the GRiDCase). These sockets are referred to as the internal ROM sockets. Each of the ROM sockets (internal and external) can have a 128 KB masked ROM, or a 64 KB (or smaller capacity) EPROM inserted into it. (With a masked ROM, a master is prepared and sent to be duplicated in mass quantity. Masked ROMs have 28 pins.) The four internal ROM sockets of the GRiDCase Plus can accomodate 32-pin 128 KB EPROM's, as well as 28-pin 64 KB EPROM's and 128 KB masked ROMs. The sizes and types of ROMs can be mixed, and arranged in any order in the sockets.

The GRiDCase has the capability of having RAM installed up to a limit of 640 KB. Under GRiD-OS, the upper area of RAM between 512 KB and 640 KB is saved for ROMs and RAM disks. Under MS-DOS, this upper RAM is used as part of main memory to be compatible with the IBM PC.

ROMs in the GRiDCase may contain one of three types of code: bootable, draggable or executable. Bootable code is invoked from the boot PROM during initialization at system power-up or system reset. The boot PROM does not occupy one of the ROM sockets, it is internal to the GRiDCase. The GRiD user does not have access to the boot PROM, and would never need to create a boot PROM. Understanding that there is a boot PROM and its function is all the information that is needed.

Draggable code is invoked by the user selecting a file for execution from ROM. Draggable ROMs are available under GRiD-OS and MS-DOS. The code in a draggable ROM is transferred from the ROM memory space to main RAM before it is executed. You can think of draggable ROMs as another media storage device, which is read-only. The ROMs created with the MS-DOS ROMBuild software are draggable ROMs.

Executable code is invoked by the user selecting a file for execution from ROM. Executable ROMs are only available under GRiD-OS or InteGRiD. The main advantage to executable code in ROMs is that it does not occupy as much of main RAM as draggable code in ROM. Instead of dragging the application into RAM and then executing, the code executes from the ROM. Therefore, when an application ROM is inserted into your system, you effectively add the RAM the application would normally require to the total RAM in your system.

Chapter 3: Compass ROM Capability

The GRiD Compass computer has four 28-pin ROM sockets accessable under a small panel located above the keyboard. Each of these sockets can contain one 128 KB masked ROM, or one 64 KB (or smaller capacity) EPROM. (As of the writing of this manual, there are no 28 pin, 128 KB EPROMs available that will work in the Compass.) The sizes and types of ROMs can be mixed, and can be arranged in any order in the sockets.

ROMs in the Compass may contain one of three types of code: bootable, draggable or executable. Bootable code is invoked from the boot PROM during initialization at system power-up or system reset. The boot PROM does not occupy one of the external ROM sockets, it is internal to the Compass. The GRiD user does not have access to the boot PROM, and would never need to create a boot PROM. Understanding that there is a boot PROM and its function is all the information that is needed.

Draggable code is invoked by the user selecting a file for execution from ROM. Draggable ROMs are available under GRiD-OS and MS-DOS. The code in a draggable ROM is transferred from the ROM memory space to main RAM before it is executed. You can think of draggable ROMs as another media storage device, which is read-only. The ROMs created with the MS-DOS ROMBuild software are draggable ROMs.

Executable code is invoked by the user selecting a file for execution from ROM. Executable ROMs are only available under GRiD-OS. The main advantage to executable code in ROMs is that it does not occupy as much of main RAM as draggable code in ROM. Instead of dragging the application into RAM and then executing, the code executes from the ROM. Therefore, when an application ROM is inserted into your system, you effectively add the RAM the application would normally require to the total RAM in your system.

Chapter 4: Creating MS-DOS ROMs

The maximum file size in ROM under MS-DOS on a GRiDCase is 1 MB. On a Compass running MS-DOS, the maximum file size is 128 KB. The software used to create MS-DOS ROMs limits the number of files to 512. There is an overhead that reduces the total amount of space available in a ROM set that is comprised of:

```
Boot sector =                                          512 bytes
Header =                                 # ROM groups * 512 bytes[1]
FAT sectors =  (((# files * 3 / 2) + 511) / 512) * 512 bytes
Sector for each 16 files = ((# files + 15) / 16) * 512 bytes
```

[1] If the total ROM size is 128K or less, then the header only occupies 512 bytes.

For the GRiDCase, the directory entries for the files in ROM are appended to the directory for drive A under version 2.11. This is transparent to the user and can lead to confusion, since there is no division between the actual directory for drive A, and the directory for the files in ROM. With version 3.2 of MS-DOS on the GRiDCase, the MODE command can be used to append the directory entries to drive A or drive B, or disable them entirely. For the Compass, the directory entries for the files in ROM are appended to the first MS-DOS mass storage device that the initialization sequence encounters. (Normally this is drive C, the hard disk). ROM files on a hard disk version of the GRiDCase Plus are appended to the hard disk (drive C) if there are no floppy drives attached. Files in ROM are always appended to the root directory, no sub-directories are allowed.

Creating ROMs

The process for creating a ROM under MS-DOS involves four steps:

1) Selecting the files to go into ROM

2) Using ROM Build to create the hex file(s)
3) Transferring the hex file(s) to the PROM Programmer
4) Programming PROMs with the PROM Programmer.

A hex file is an ASCII file that contains the hexadecimal character equivalents for the bytes in a data file. Hex files created with GRiD's ROM Build software are patterned after the Intel hex format. The ROM Build utility is described in detail later in this chapter. If the files being put in ROM are from a commercially available application, it may be necessary to do some preparation work on the files before you use ROM Build.

Commercially Available Applications

It is possible to take an application that you have purchased commercially and put it in ROM. Care must be taken to do it correctly, or the application may not work. The four areas to examine are installation/configuration, extraneous files, copy-protected software, and read/write access of files. Each of these is described below.

Installation/Configuration

Many applications require you to install them on a diskette or hard disk drive. In this process you usually configure certain system-related parameters, such as what printer you are using, the type of monitor you have, and even what plotter you may have attached. Before you put the application into ROM, you must go through this configuration process, because once it is in ROM it cannot be changed.

It is best to follow the manufacturer's instructions and install the application on a diskette (or hard disk). Be sure and keep track of the files that are written to disk. Go ahead and configure the application per the instructions the manufacturer has given you. You are now ready to go on to the next step which is eliminating unnecessary files.

Extraneous Files

Most applications have files that pertain to their installation and configuration that are not necessary to put in ROM. It is often necessary to eliminate these files so that the application will fit into a reasonable number of ROMs. These files will have names such as CONFIGUR.DAT, READ.ME, MONITOR.DAT, PRINTER.DAT, HELP.DAT, INSTALL.EXE, etc. There may also be sample data files with names like SAMPLE.TXT, BEGIN.DAT, etc.

With the application installed, you should start renaming these files one at a time (to a dummy file name) and then test the application to see if it runs without the eliminated files. Be sure to test all of the applicable features of the renamed file before you decide to eliminate the file. If the application works, the file you just renamed is probably not necessary. If the application fails, rename that file back to its correct name. Continue this process until you have pared the application down to a reasonable size. Make sure you remember to restore

the renamed files to their original names.

Copy Protection

Many applications are copy-protected by the manufacturer. This can present problems when trying to put the application in ROM. With the application installed on a diskette, and the extraneous files eliminated, write protect the diskette, and then try the application. Usually if the application works from a write-protected diskette, it will work in ROM. There are occasions where it will not. The way to set the files to read-only is by using the ATTRIB command under MS-DOS version 3.2. This is a very effective way of testing an application to see if it will work in ROM.

If the program does not work, it will be necessary to acquire or produce a version that is not copy-protected to download into ROM. Various software techniques exist in the market to produce versions that are not copy-protected. GRiD does not supply this software. Users should check with their own legal counsel on these matters.

If you intend to make multiple ROM copies of an application, you should contact the manufacturer and discuss what is called a site-license. They will sometimes supply you with a version of their application that is not copy-protected for you to put in ROM.

Read/Write Access

Many applications open files for read/write access even though they only need to read them (such as the case of a static configuration file). Opening a file in ROM for read/write access will give an error since each file placed in ROM is flagged as being read-only. Opening a file on a write-protected diskette for read/write access will only give an error when the file is written to. This is important to remember when moving an application from disk to ROM, or developing an application for use in ROM.

MS-DOS ROM Building Software

ROM Build is used to perform a three step process: 1) Select the files to be input into ROM, 2) Build those files into a hex image, and 3) Transfer the resultant hex files to a serial PROM programmer. If a PROM programmer with its own software interface is being used, i.e., the Sailor-8, then the software supplied with the PROM Programmer would be used to transfer the hex files, built by ROM Build, to the programmer. Therefore, step 3 would be eliminated.

ROMBUILD.EXE

ROM Build is a menu-driven utility that allows users to select files to be placed in ROM, and then transfer the resultant hex file(s) to a PROM programmer attached to the serial port. It keeps the user informed as to

how much space is available in the ROM, and forces the user to correctly fill in the required fields to build a ROM. The main menu of ROM Build is shown in Figure 4-1. The arrow keys move the highlighted rectangle from one selection to another, and the <RETURN> key is used to confirm the entire menu.

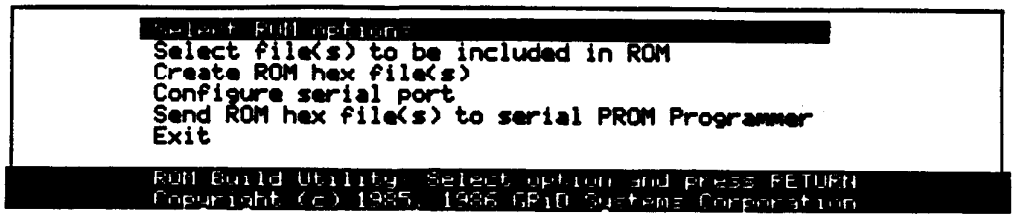

Figure 4-1. ROM Build Main Menu

Select ROM Options

The first step in the process of putting files in ROM is to set up the options for the ROM. The options menu of ROM Build is shown in Figure 4-2. Certain fields are optional as noted.

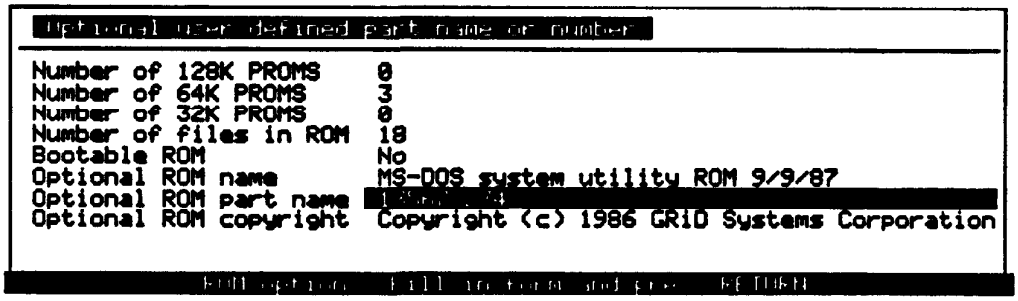

Figure 4-2. ROM Build Options Menu

The first three items in the menu specify how many and which type of PROMs are being used. Any combination of PROMs may be used.

The specification of the number of files is used to calculate the number of directory sectors. This number is rounded up to the nearest multiple of sixteen.

The ROM name, ROM part name (part number) and ROM copyright items are fields that are placed in the ROM identification area.

If you specify a bootable ROM, ROM Build automatically builds in the boot sector and displays the menu shown in Figure 4-3. You must specify which system (GRiDCase or Compass) the ROM is to be used in, since the boot sectors are different for the two systems (the GRiDCase is PC

compatible). You can optionally specify that the COMMAND.COM file and the invisible files necessary to boot MS-DOS be included. These system files are not necessary to make a bootable ROM for applications that use their own operating system, they are only required for MS-DOS. If you specify that these files are to be included, you will be prompted to insert a system diskette in the specified drive, and a warning will be displayed informing you how much space those files require. For MS-DOS version 2.11 this is almost 58 KB. For MS-DOS version 3.2 this is 66 KB (note that MS-DOS version 3.2 will not fit in one 64 KB ROM). A boot prompt key is also solicited (1, 2, 3, or 4). The boot prompt key is the numeric key to be pressed to boot this ROM (useful if more than one bootable ROM is installed). This menu is shown in Figure 4-3.

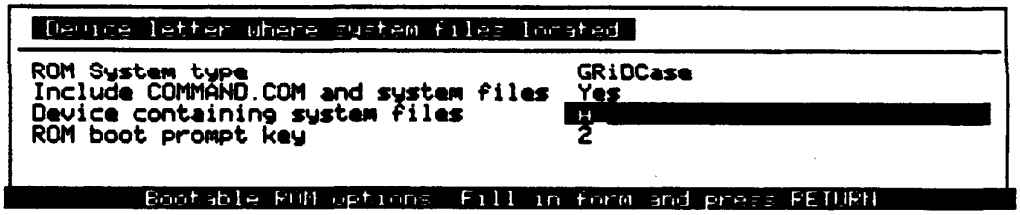

Figure 4-3. ROM Build Boot Menu

Once the ROM option menu has been filled in, the next menu that appears asks for the destination hex file name. Fill in the desired pathname for the hex files to be generated by ROM Build. The file extension HX1, HX2, etc. is appended to each hex file created. Make sure there is enough room on the hard disk for the hex files to be stored. For 64KB and 128 KB ROM hex files, 157 KB and 315 KB of disk space is required respectively for each hex file.

Select file(s) to be included in ROM

To select a file for inclusion in ROM, you can select a drive from the file form, or a particular file from the currently selected drive. (The file form is what is displayed on the screen when the Select File(s) to be Included in ROM option is chosen from the main menu.) Wildcards are allowed.

Files that are hidden from the directory can also be included in ROM. The pathname for each hidden file must be typed in as these filenames are not visible in the directory. (Hidden files will not be included when wildcards are specified.) The file selection menu is shown in Figure 4-4.

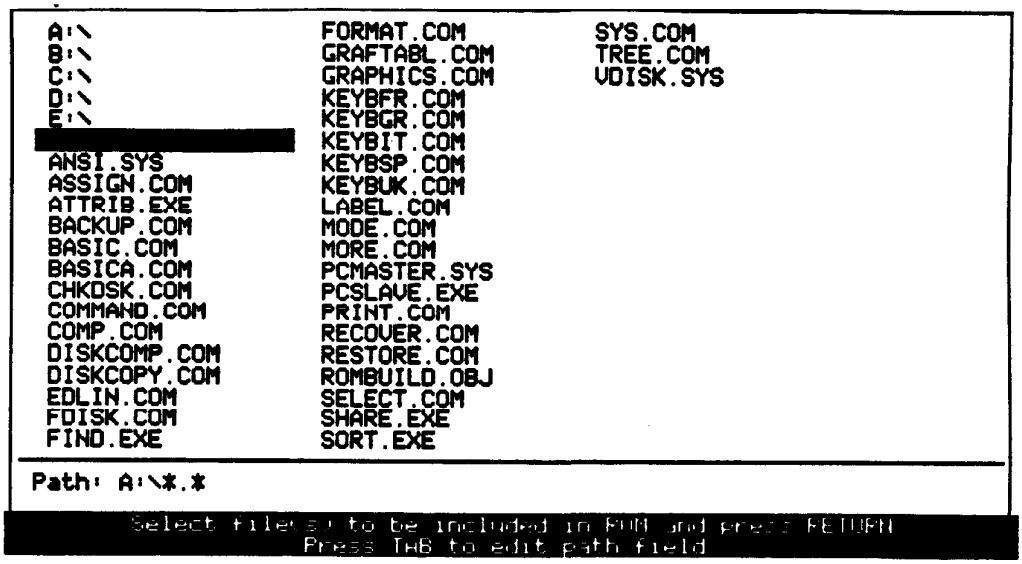

Figure 4-4. ROM Build File Select Menu

As you select files to be included in the ROM, ROM Build displays each filename, its size and last-modified date and time, the amount of space left in the ROM, and the files already included. You then have the option of including the file or not. A sample of this verification form is shown in Figure 4-5.

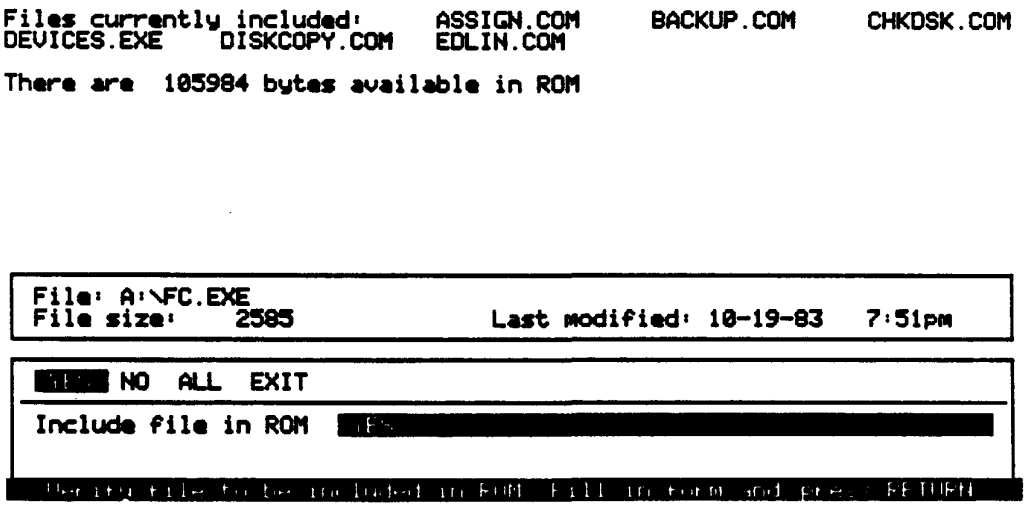

Figure 4-5. ROM Build Verify Include File Form

The ALL and EXIT options are only displayed if you have specified a wildcard in the filename. ALL indicates that all files matching the

wildcard specification are to be included. EXIT allows you to end the inclusion of files and return to the main menu where you can continue selecting files or build the hex file.

Create ROM hex files

Once you establish the desired list of files, you would then build the hex files by selecting the Create ROM Hex File(s) choice from the ROM Build Main menu. The checksums for each hex file are displayed at this point. You should copy them down for future reference against the checksums of the resulting PROMs programmed on your PROM programmer. Once all the hex files are built, you can transfer them to the PROM programmer through the serial port. If you are using the Sailor-8 PROM programmer, you should now exit from ROM Build and use the software that accompanies the Sailor-8 (see Appendix C).

Configure serial port

A configuration menu is available to initialize the serial port; to set the baud rate, data bits and stop bits values that are necessary to communicate to your serial PROM programmer. At this point the user should prepare the PROM Programmer as instructed by the programmer's reference manual. If the DATA I/O PROM programmer is being used, refer to Appendix C.

Send ROM hex file(s) to serial PROM Programmer

Once the serial port has been configured and the PROM programmer is ready to receive data, select the main menu choice of "Send ROM hex file(s) to serial PROM Programmer". Fill in the form with the hex files that are to be sent to the programmer. As each hex file is sent, a percent complete message is displayed for your convenience. When the transfer of the hex file is complete, a screen is displayed as a reminder to label the PROMs, verify the checksums, etc.

Chapter 5: Creating GRiD-OS ROMs

The maximum file size in ROM under GRiD-OS is 128 KB. There is a practical limit to the number of files you can place in a 128 KB ROM of 2026. The reason is that the smallest allocation unit for GRiD-OS files is 256 bytes (the size of a sector). There is an overhead that reduces the total amount of space available in a ROM set that is comprised of:

```
Sector 0 =                                                    256 bytes
Header =                                                      256 bytes
Sector per file =                                (# files) * 256 bytes
Sector per file > 5 KB =                  (# files > 5 KB) * 256 bytes
Sector per file > 70 KB =                (# files > 70 KB) * 256 bytes
Directory = ((Len file names + (3 * # files)) / 256 + 1) * 256 bytes
```

Files in ROM under GRiD-OS can be executable from ROM or draggable (loaded into main memory). Draggable and executable files can be mixed within one ROM set. The directory entries for the files in ROM appear in a device called "Read Only Memory", in the subject "Programs".

Occasionally it is necessary to transfer GRiD-OS hex files to MS-DOS media via GRiDTransfer in order to take advantage of download software specific to the PROM programmer you are using (such as a One-D Sailor 8 programmer). Many manufacturers of PROM programmers have taken an MS-DOS software-interface approach to communicating with their programmers.

Creating a ROM set involves selecting the files and calculating the space required for them. By using the formula above to calculate the overhead, and knowing that a file occupies a minimum of one sector (256 bytes), you can determine the size of ROM space necessary for the group of files you want to put in ROM. You then proceed with ROM Builder-Task~ which is described later in this chapter. The program will build the hex files and optionally send them through a "termulator" (terminal emulator) to a PROM programmer.

In certain instances, you may have to use ROM Builder~Run~ and PROM~Run~ directly instead of using the ROM Builder~Task~ interface to create the hex files. Both of those utilities are described later in this chapter as well.

If you have a PROM programmer that requires a parallel interface or a PC-based MS-DOS interface, you should not use the termulators; instead use GRiDTransfer to copy the hex files to MS-DOS media on a PC or GRiDCase, where you can use MS-DOS software to download them to your PROM programmer.

GRiD-OS ROM Software

Transferring GRiD-OS files into ROM is a four step process:

1) Combining the desired files into a ROM image
2) Translating the ROM image into an Intel hex format file
3) Transferring the hex file to the PROM programmer
4) Programming the PROM using a PROM Programmer

This process is illustrated in Figure 5-1.

Occasionally it is necessary to transfer GRiD-OS hex files to an MS-DOS system via GRiDTransfer in order to take advantage of download software specific to the PROM programmer you are using (such as a One-D Sailor 8 programmer). Many manufacturers of programmers have taken a software-based programmer control approach, and have selected MS-DOS as the vehicle.

In addition to the software that is specific to building ROMs, there are a variety of other utilities supplied by GRiD that are useful in preparing files for ROM building. They are documented elsewhere, and are mentioned here just to establish their existence.

GRiDWrite Text editor used to manipulate map files from LINK-86 and output files from ROM Builder.

GRiDFile Database application used to sort data from map files.

GRiDDevelop Menu driver interface to the development process. Invokes compilers, linkers and other utilities needed when developing software.

GRiDTerm Termulator used to transfer files to PROM programmers.

GRiDTransfer Utility to transfer files from GRiD-OS to MS-DOS. This step is necessary if the PROM programmer you are using has an MS-DOS programmatic interface (such as the Sailor-8). Works from one media to another media (not across serial ports).

Note: To transfer GRiD-OS files to MS-DOS with

GRiDTransfer, select the option, GRiD-OS to MS-DOS - No Conversion from the GRiDTransfer menu.

The utilities described in the following sections are provided for use with the ROM Build software and for the convenience of the application programmer. They are not supported as individual products.

ROM Builder~Task~

The ROM Builder task program is a menu and form driven interface to the ROM Builder~Run~ and PROM~Run~ programs, as well as the termulator connection to the PROM programmer or to GRiDTransfer. It leads the user through the necessary steps to place files in ROM. The maximum size of the ROM package is 128 KB. ROM Builder~task~ requires 512 KB of memory, GRiDTask (or GRiDTask II), and a hard disk to execute.

The ROM Builder task program is intended for use by less demanding application developers who only want to put files into a ROM. It does not produce bootable ROMs, nor does it allow designating a part number.

The main menu is displayed as follows:

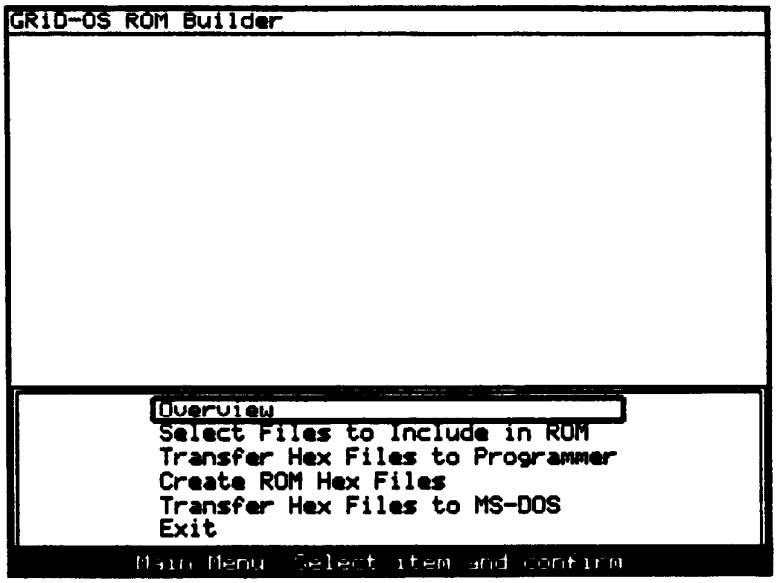

Figure 5-2. ROM Builder Main Menu

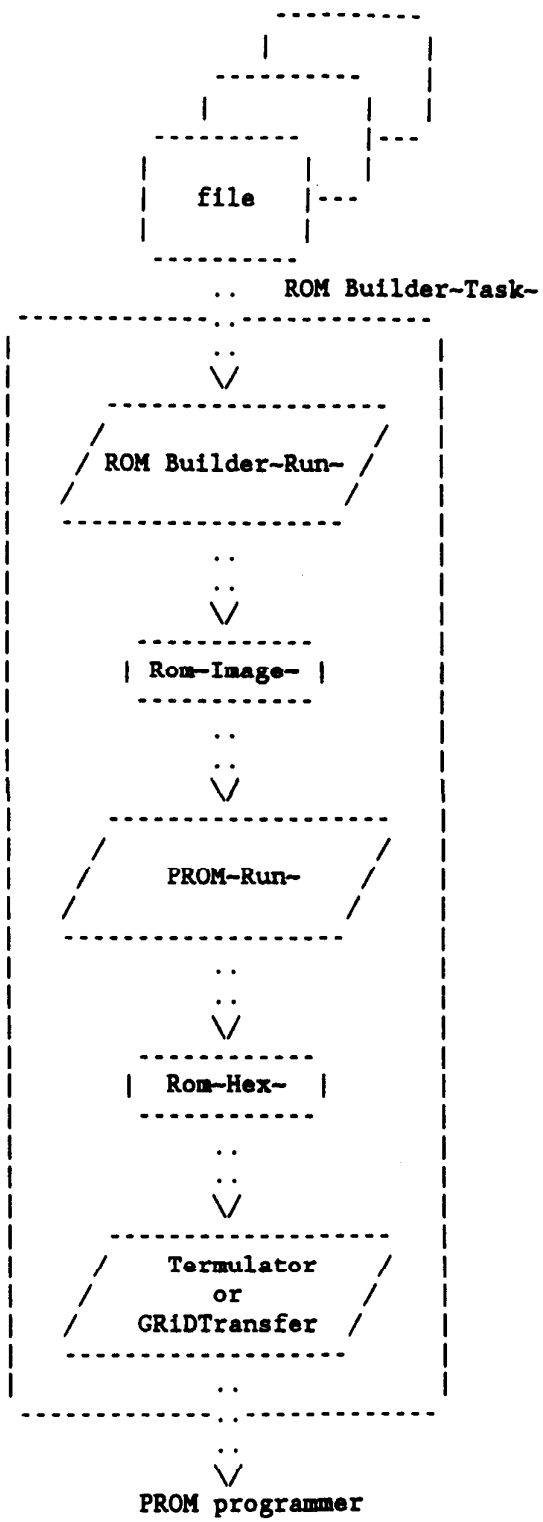

Figure 5-1. PROM Programming Process

The ROM Builder task program always uses the same names for ROM image and ROM hex files (ROM~IMAGE~ and ROM~HEXn~, where "n" is a single digit). These files will be placed in the device and subject from which ROM Builder is executed. If you repeat the ROM Builder task program, the previous image and hex files will be erased and over-written.

The ROM Builder task program is designed to be self-explanatory, so no detailed description of it is included here. If you wish to modify the task program, the source code has been included on the diskette accompanying this manual. The following two sections describe in detail the programs that are executed by the ROM Builder task program. When executing ROM Builder task, it is important to pay very close attention to the other utilities that it is managing. Errors within the termulators, GRiDTransfer, etc. are not trapped by the ROM Builder task. The user is responsible for noting any errors and taking corrective action.

ROM Builder~Run~

Depending on your expertise and the situation, it may be more effective to forego the ROM Builder task program and incorporate the necessary ROM building steps into a GRiDDevelop file for the application. This allows specification of all of the options listed below (ROM Builder~Task~ allows a pre-determined subset). This step can be complicated, and should only be taken if you are comfortable working with GRiDDevelop files. The command invocation for ROM Builder is:

```
ROM Builder {filespec} [,{filespec}]... TO {image pathname}
[options]
```

where: {filespec} is "{pathname} [({segname}[,{segname}]...)]", the optional {segname} parameter designates the code groups of an executable file. ("CGROUP" for a single subsystem application.)

{image pathname} designates the destination file of kind ~Image~.

[options] include:

RomID({number})	5001 through 32767
Part('{string}')	9 characters ("nnnnnn-nn")
RomSize({number})	32, 64, 128
WaitStates({number})	must be set to three
NumRoms({number})	1 through 8
SystemRom	indicates a bootable ROM
Invisible	non-file system binary file
Print({pathname})	redirects screen output to a file

Options are separated by spaces. For more details on the first six options, see Appendix D. The invisible option specifies that the files being placed in ROM are not to be added to the file system. The application is responsible for manipulating this data. All files placed

in ROM are in the Programs subject.

PROM~Run~

The PROM program takes the ROM~image~ file that is generated by ROM Builder~Run~ and creates the ROM~hex~ file(s) necessary to burn the PROMs themselves. For a 64 KB ROM, the hex file created occupies 184 KB on the disk. Be sure you have enough disk storage before beginning the hex file creation process. The command invocation is:

PROM {image pathname} ({number})

where: {image pathname} is the same name as the output file from ROM Builder

{number} is the size in KB of the PROMs to be programmed (32, 64 or 128)

The PROM program generates one or more hex files depending on the size of the image file. The output files have the same title as the image file, but have a kind of ~HEX1~, ~HEX2~, etc. PROM displays the checksums of the original image file and the resultant hex file. If these two checksums are not equivalent, an error has occurred, you should check your work, and repeat the process.

Wavetek~Terminal~

This is a termulator configuration file used with GRiDTerm to communicate hex files to a Wavetek 824 PROM programmer. It is invoked from within ROM Builder~Task~. It is set up for 9600 baud, 8 data bits, 1 stop bit, full duplex, no parity. It could be used for other PROM programmers that have a similar serial interface.

DataI/O~Terminal~

This is a termulator configuration file used with GRiDTerm to communicate hex files to a Data I/O 121 or 29A PROM programmer. It is invoked from within ROM Builder~Task~. It is set up for 9600 baud, 8 data bits, 2 stop bits, full duplex, no parity. It could be used for other PROM programmers that have a similar serial interface. See Appendix C for more information on the Data I/O programmer.

Other~Terminal~

This is a termulator configuration file used with GRiDTerm to communicate hex files to a generic PROM programmer. It is invoked from within ROM Builder~Task~. It is set up for 9600 baud, 8 data bits, 1 stop bit, full duplex, no parity.

Appendix A - Handling EPROMs

Static Damage

EPROMs, like all integrated circuits, are susceptable to damage from static electricity. Always ground yourself before handling them. To further eliminate any possibility of damaging the EPROMs, handle only from the ends, without touching the leads (or "legs").

The EPROM programmer operator should be grounded at all times with a wrist strap. This wrist strap should be tied to an earth ground through a one-megohm resistor. Wrist straps are available from CharlesWater, part number CP405. CharlesWater can be reached at (617)964-8370 (East Coast), (818)502-1453 (West Coast) or (0892)31012 (England).

The EPROM programmer work space should be grounded with a static control table and/or floor mat that is connected to an earth ground. These grounding mats are also available from CharlesWater, part numbers CP604 and CP603 respectively.

Anti-static containers should be used to house the EPROMs both before and after programming. This is true of both crimped and uncrimped EPROMs.

Ultraviolet Light Damage

EPROMs are also susceptable to damage from ultraviolet light after they are programmed. EPROMs are erased with a high-intensity UV light, but can be damaged (partially erased) by lower intensity UV light, such as sunlight or light from flourescent fixtures. Make sure that you always afix a label to the EPROM that completely covers the window.

Erasing EPROMs

There are special devices made for erasing EPROMs that emit high-intensity ultraviolet light. Before inserting EPROMs into the chamber, make sure you have completely removed the label and any glue residue from the window. When you place the EPROMs in the chamber, make sure that the window is in full view of the UV light source. Depending upon the manufacturers instructions, you should leave the EPROMs in the chamber for 20 to 40 minutes. EPROMs that have been programmed and erased several times may require a full 60 minutes to erase. Be careful not to over-erase.

Diagnosing Failed EPROMs

Using a PROM programmer, blank check the EPROM. If the check fails, follow the directions above for erasing it, and blank check it again. If the check still fails, the EPROM is bad. If the blank check is successful, program the EPROM. If programming fails at byte 0, the EPROM is bad. If the programming goes to completion, and upon examining the EPROM's contents you find all the bytes the same, the EPROM is bad. If the same EPROM consistently comes up with a bad checksum, then the EPROM is bad. Failed EPROMs, if new, should be returned to the distributor for replacement. Failed EPROMs, if re-used several times, should be discarded.

Appendix B - GRiD Part Numbers for EPROMs

QUANT	PART DESCRIPTION	PART #
1	Crimped 64KB EPROM	24200
15	Crimped 64KB EPROMs	24202
1	Uncrimped 64KB EPROM	24201
15	Uncrimped 64KB EPROMs	24203
1	Uncrimped 128KB EPROM	24204
10	Uncrimped 128KB EPROMs	24205
5	Molex sockets (for crimped 28 pin ROMs)	24210
10	ROM carriers (to crimp 28 pin ROMs into)	24211
1	Internal 28 pin ROM board (GRiDCase)	101412

Appendix C - PROM programmers

A variety of PROM programmers are commercially available. This appendix provides some background information on generic serial programmers, and some helpful information on three specific PROM programmers. The appendix is outlined as follows:

1) Generic serial programmers
2) DATA I/O programmer
3) Sailor-8 programmer
4) EPRO programmer

Generic serial programmers

The term generic is intended to mean that the PROM programmer being used does not require any programmatic interface. The most common kinds of PROM programmers include the Data I/O 29A, Data I/O 121, Prolog 912A, and the Wavetek 824. These PROM programmers expect to receive an Intel hex file via the serial port. All other set-up or configuration data is entered through a control panel on the PROM programmer itself.

You communicate with serial PROM programmers using termulators under GRiD-OS. With the termulators, you can create terminal data files with the exact parameters for your programmer (such as the DataI/O-Terminal- and Wavetek-Terminal- files described in Chapter 5). Under MS-DOS, ROMBUILD has a "Send ROM hex file(s) to PROM programmer" option to transfer hex files to your PROM programmer.

C-2 Programming EPROMs for GRiDCase and Compass

DATA I/O PROM programmer

The following is a step by step procedure that should be followed to burn EPROMs on the DATA I/O Serial PROM programmer.

1. Build hex file(s) using the ROMBuild program.

2. Prepare to transfer files to the DATA I/O PROM programmer by selecting the "Configure serial port" option on the main menu and configuring the serial port as follows:

   BAUD = 9600
   Parity = None
   Stop Bits = 2
   Data Bits = 8

3. Prepare the DATA I/O PROM programmer as follows:
   a) Connect the PROM programmer cable to the serial port on your system
   b) Turn on the PROM programmer
   c) At the PROM programmer, press the following keys in the EXACT order listed:

      SELECT 1 4 SHIFT2-BACKSPACE START 0 START

4. Select the "Send ROM hex file(s) to serial PROM programmer" selection at the main menu and select a hex file (or files using wild cards) to send hex files to the PROM programmer. The file to be sent to the PROM programmer is displayed and a message prompts the user to press RETURN to send the hex file to the PROM programmer.

   Before sending the HEX file into the PROM programmer, enter the following commands at the DATA I/O PROM programmer:

      LOAD LOAD START 8 3 START

   Press RETURN at the system keyboard to begin sending the hex file to the PROM programmer. Watch for the clock to turn on the PROM programmer display. If the clock does not turn, data is not being transmitted and the connection between your system and the PROM programmer should be checked.

5. Programming a PROM at the PROM programmer:
   Start programming the PROM by entering the following commands on the PROM programmer:

   PROG START partSequence START'

   the part Sequence for part number AM27512:

      SHIFT3-7 1 SHIFT-7 4

   the partSequence for part number AM27256:

SHIFT3-7 1 3 2'

Insert a PROM into the PROM programmer, and press START START on the PROM programmer.

NOTE: Programming should begin. Once again, the clock on the PROM programmer display should turn. If the clock does not turn, reenter the commands listed above on the PROM programmer making sure the part sequence number was entered correctly.

6. Repeat steps 5 and 6 to continue sending hex files to the DATA I/O PROM programmer.

7. After all hex files have been sent to the PROM programmer, and after all the PROMs have been programmed:

   1) Remove the PROM from the PROM programmer
   2) Power off the system
   3) Insert PROMs into your system
   4) Turn on your system and verify the PROM programming is correct
   5) Label each PROM with date, checksum and version number

Sailor-8 PROM programmer

The Sailor-8 PROM programmer from the ONE/D Corporation allows users to program a wide variety of EPROMs. The Sailor-8 programmer will program both 28 pin 64 KB and 32 pin 128 KB EPROMs. An adapter card must be purchased in addition to the Sailor-8 to support the 32 pin EPROMs. Contact ONE/D for more information at:

ONE/D
1050 L East Duane Ave.
Sunnyvale, CA 94086

(408)969-9900

The Sailor-8 has a parallel communications interface which offers a higher speed of transfer than the serial interface. Instead of a control panel user interface, the Sailor-8 uses a programmatic user interface that runs under MS-DOS. It utilizes Lotus 1-2-3 style menus for interactive operation, and also has a macro interface for repetitive operations. It requires a minimum of 256 KB of memory. It can program a single PROM, a set of PROMs or a "gang" of the same PROM. You can download a hex file through the parallel interface and then program PROMs from the buffer, or you can copy a master PROM into the buffer, and then program PROMs from that.

A few things to be aware of when operating the Sailor-8:

- Do not turn the Sailor-8 on or off with PROMs in the sockets.
- Connect the GRiDCase to the Sailor-8. Power on the GRiDCase, then the Sailor-8.
- Insert PROMs with the notch towards the top (back) of the Sailor-8.
- If you have purchased your PROMs from GRiD, make sure you use chip type #7 in the configuration menu. This is an Intel or AMD 27512 (64 KB) EPROM that expects a programming voltage of 12.5 volts. Chip type #8 is also a 27512, but will destroy the GRiD-supplied 27512s because it uses a different voltage (20 volts). If you have purchased your PROMs from a source other than GRiD, make sure you configure the Sailor-8 correctly for the chip type you have. Programming of the PROMs may appear to work, but if the proper personality code isn't used, problems are likely to develop.

The main menu screen of the Sailor-8 software offers the following options:

```
Configure
File
Buffer
Release-Control
PROM
Macro
```

Use the Configure option to change the following defaults:

Field Name	Default	Correct Setting
Port	LPT1	LPT2
Chip-Type	8K PROM	64K PROM, 12.5 V
Set-Size	8	1

Once the configuration settings have been changed to their proper settings, follow the next five steps to program one 27512K (64KB) EPROM: (Refer to the Sailor-8 documentation for more complex operations.)

1. Use the **File/Name** option to enter the filename of the hex file you want to program into PROM. This is the file that was output from GRiD's ROMBuild software.
2. Escape back to the main menu. Use the Buffer/Load/0 option to load the hex file into buffer zero.
3. Escape back to the main menu. Use the Prom/Blank/0 option to verify the PROM you want to program is blank. (Make sure when inserting the PROM into the programmer that the notch is on the top end of the chip).
4. Use the Prom/Program/0 option to begin programming your PROM.

5. Once the PROM is programmed, verify the checksum on the PROM with the Prom/Checksum/0 option. This checksum should match the checksum of the hex file that was given to you from the ROMBuild software when the hex file was created.

   Note: When programming more than one buffer (such as is necessary for a set of ROMs), you must load each hex file to buffer zero and do a "Buffer/Duplicate" command to transfer the data to the desired buffer. Make sure the set size has been set to one before attempting this operation.

   The manual that accompanies the Sailor-8 more fully describes the other commands, options and macro capabilities that are available.

EPRO PROM programmer

EPRO Corporation offers the EPRO 4000, a PROM programmer which allows users to make master 128KB 32 pin EPROMs. These EPROMs can only be used in GRiD computers which have the capability of handling the 32 pin EPROMs. The EPRO 4000 also allows duplication of 128KB EPROMs, one PROM at a time.

The EPRO 4000 comes with a programmer, an expansion card (to be installed inside a GRiD Expansion Box or an IBM PC), and software. The EPRO software requires an Intel Hex file in order to make a PROM. Therefore, customers must first use GRiD's EPROM software (GRiD-OS or MS-DOS) to create the Hex file before using the EPRO Programmer. The EPRO software loads the Hex file into memory, manipulates the data, and burns the result into the EPROM in the programmer.

For more information, contact EPRO at:

EPRO Corporation
2342 Harris Way
San Jose, CA 95131

(800)433-EPRO
(800)262-EPRO
(408)433-5555

Appendix D - ROM Structure

The information in this appendix is included for the more technically oriented user of the EPROM Programming Kit. If even more technical information is required, contact your GRiD Representative.

The ROM header is always located in the last 256 bytes of the ROM. When the ROM is mapped in, the header is located at segment 9FF0h in memory, regardless of the size of the ROM. A 128 KB ROM starts at memory location 8000h (just above the 512KB of system RAM). A single 64 KB ROM starts at memory location 9000h. A single 32 KB ROM starts at memory location 9800h. A diagram of ROM mapping is given in Figure D-1.

It is possible for one set of files to span multiple ROMs. In the case of the Compass and the GRiDCase external ROM sockets, 64 KB EPROMs are the largest capacity EPROMs that can be used. Two 64 KB EPROMs (or four 32 KB EPROMs) can be mapped into memory as a package, occupying the entire 128 KB of memory available for ROMs. That mapping is shown in Figure D-2.

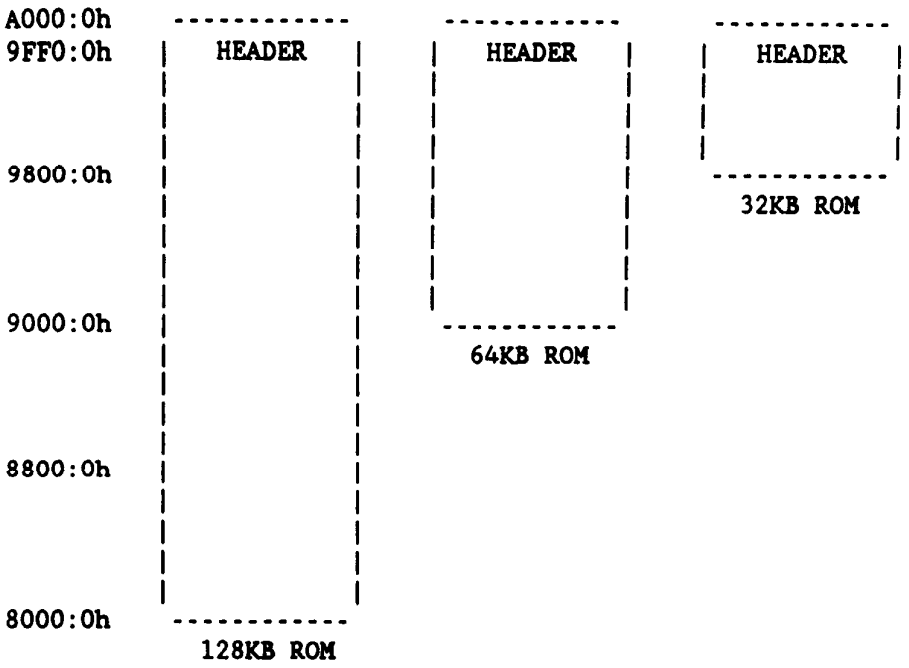

Figure D-1. Memory Mapping of a Single ROM

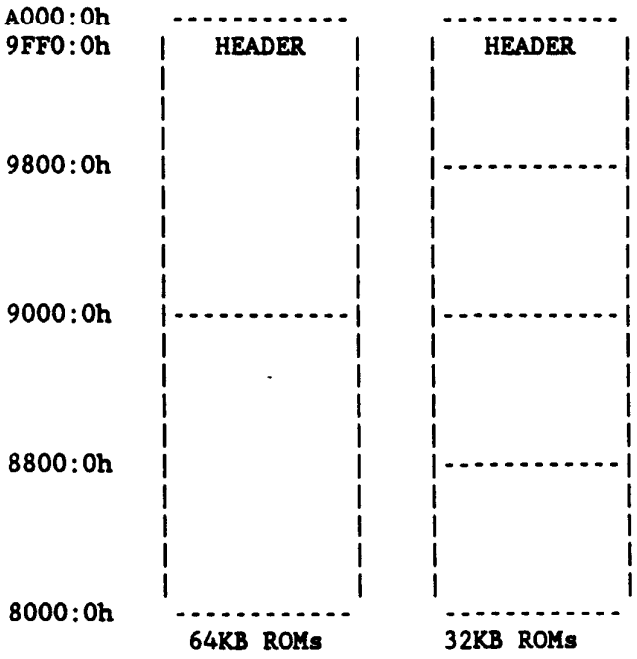

Figure D-2. Memory Mapping of a ROM Package

Note: There is only one ROM header per package.

A single large file or group of files that exceeds 128 KB is stored in multiple ROMs, known as a group. Each ROM in the group has its own

header. Groups are available under MS-DOS only.

ROM Header

The ROM header contains three kinds of information: identification fields, bootstrap fields and directory fields. The organization of the data in the ROM header is given in Figure D-3. The "ID", "BOOT", and "DIR" columns indicate what kind of information is contained in the field. A more complete description of these field types is given in the three sections following this table.

ADDRESS	OFFSET	FIELD NAME	LENGTH	ID	BOOT	DIR
9FF0:0h	00h	romHereFlag	2	*		
9FF0:2h	02h	sysType	1	*		
9FF0:3h	03h	systemRom	1		*	
9FF0:4h	04h	bootSector	2		*	
9FF0:6h	06h	romAddr	1	*		
9FF0:7h	07h	romSize	1	*		
9FF0:8h	08h	waitStates	1	*		
9FF0:9h	09h	dirSelector	2			*
9FF0:Bh	0Bh	dirLength	2			*
9FF0:Dh	0Dh	numFiles	2			*
9FF0:Fh	0Fh	pageZeroSelector	2			*
9FF1:1h	11h	totNumPages	2			*
9FF1:3h	13h	romId	2	*		
9FF1:5h	15h	copyRight	45	*		
9FF4:2h	42h	time	11	*		
9FF4:Dh	4Dh	partNumber	9	*		
9FF5:6h	56h	partName	15	*		
9FF6:5h	65h	bootLength	1		*	
9FF6:6h	66h	bootMessage	30		*	
9FF8:4h	84h	bootId	1		*	
9FF8:5h	85h	numRomsInPkg	1	*		
9FF8:6h	86h	romSumsArray (4)	8	*		
9FF8:Eh	8Eh	romIdText	50	*		
9FF8:Fh	8Eh	unused	60			
9FFF:Ch	FCh	numRomsInGroup	1	*		
9FFF:Dh	FDh	groupSequenceId	1	*		
9FFF:Eh	FEh	patchCodeFlag	1	*		
9FFF:Fh	FFh	checkSum	1	*		

Figure D-3. ROM Header

ROM Identification Fields

romHereFlag	Set to BB66h to indicate presence of ROM header.
sysType	Identifies the operating system the ROM is designed for. A value of 0 indicates a GRiD-OS ROM, and a value of 1 indicates an MS-DOS ROM. Values up to 80h are reserved for GRiD use.
romAddr	Set to zero (0).
romSize	Size of ROM in 1 KB units (32, 64, 128).
waitStates	Set to three (3).
romId	A 2-byte value that uniquely identifies this ROM. Values 0 through 5000h are reserved for GRiD use.
copyRight	A 45 character ASCII field for copyright information.
time	An 11 byte field that contains a time/date stamp identifying when the ROM was made. The bytes are defined as follows: 0,1 year 2 month 3 day 4 hour 5 minute 6 seconds 7 hundredths of seconds 8 day of week (not used) 9,10 day of year (not used)
partNumber	A 9 byte ASCII field that contains the part number. GRiD part numbers are formatted as 'nnnnnn-nn'.
partName	A 15 byte ASCII field that contains " <- Part Number".
numRomsInPkg	If the total ROM size is less than 128 KB and the hex file spans multiple ROMs, then the ROMs are known as a package. This is the number of ROMs in the package.
romSumsArray	A four word array that contains the individual checksums of the ROMs in the package.
romIdText	A 50 byte string identifying this package.
numRomsInGroup	If the total ROM size is greater than 128 KB, then the ROMs are known as a group. This is the number of ROMs in the group. (MS-DOS only.)

groupSequenceId	For a group, each ROM has its own header. This field indicates the sequence number of this ROM in the group. (MS-DOS only.)
patchCodeFlag	Always false.
checkSum	A single byte checksum of all of the ROMs represented by this header (a package).

ROM Bootstrap Fields

systemRom	A value of 80h identifies a bootable ROM.
bootSector	A word value to add to 8000h to create a segment address of the boot sector.
bootLength	Length of the boot message.
bootMessage	A 30 byte ASCII message. This message will only be displayed if there is more than one bootable ROM available. In that case the user is prompted with the boot messages.
bootId	Integer number the user would enter to select between bootable ROMs. For the convenience of the user the boot message should indicate what this number is.

ROM Directory Fields

dirSelector	A word value to add to 8000h to create the segment address of the first directory sector.
dirLength	Number of paragraphs in directory.
numFiles	Number of directory entries.
pageZeroSelector	A word value to add to 8000h to create the segment address of the first data sector.
totNumPages	Number of data sectors in ROM.

Glossary

bootable	The contents of the ROM are organized such that a predefined area of the ROM is loaded into the boot space in memory and then executed when the system boots. The rest of the contents of the ROM can be draggable or executable.
checksum	A byte-wide sum of all of the bytes within a file. Used to verify that a file has been copied correctly from one device to another.
copy-protection	Many commercially available applications have had the distribution media marked in some way to prevent illegal copies from being made. There are many different schemes to copy protect software, and only some of them cause problems in ROM.
crimped	An integrated circuit (IC), such as a ROM, that has been inserted into a carrier and had its leads crimped to hold it to the carrier. Crimped ROMs are used in the external ROM slots on GRiDCase or Compass.
draggable	The contents of the ROM are organized as files with directory entries. They can be read into memory and executed there. The files are read only, and cannot be modified until they are in memory.
EPROM	Erasable Programmable Read-Only Memory. The IC can be erased and re-used.
executable	The contents of the ROM are organized as executable

	files. They are executed within the ROM's memory space, and cannot be modified at all.
external	The ROM sockets located above the keyboard under a small metal cover on both the Compass and GRiDCase.
FAT	File Allocation Table. An area of an MS-DOS disk device that contains mapping information about where the sectors that compose files are stored.
gang	A set of PROMs to be programmed all with the same data on a PROM programmer.
group	A feature under MS-DOS that allows file(s) to span the 128 KB boundary of ROM memory space by occupying more than one ROM. Each ROM in the group has its own header.
header	A 256 byte area reserved at the top of ROM memory space (offset 9FF0h) that contains information about the ROM and its contents.
hex file	An ASCII file that contains the hexadecimal character equivalents for the bytes in a data file. Intel format hex files.
internal	The ROM sockets located inside the GRiDCase enclosure on a separate printed circuit board.
KB	Abbreviation for kilobyte (1024 bytes).
masked ROM	A photo-resist masking process used in making a programmed memory device.
MB	Abbreviation for megabyte (1,048,576 bytes).
package	A set of ROMs mapped together into the 128 KB ROM memory space. Only one ROM header exists for a package.
paragraph	16 bytes of memory.
PROM	Programmable Read-Only Memory. The IC is a masked part and cannot be erased or changed.
ROM	Read-Only Memory. Used as a generic designation for PROMs and EPROMs in this manual when no differentiation is necessary between erasable and non-erasable parts.
ROM image	An intermediate file in the process of creating a hex file. Output from RomBuilder-Run~.
sector	Under MS-DOS this is 512 bytes, under GRiD-OS this is 256 bytes.

set	A group of ROMs used together as a single unit. Mapped together in one contiguous memory space.
site-license	A relatively new concept in licensing software, where one company has the right to make multiple copies of a licensed application within one locale.
socket	A plastic receptacle located on a printed circuit board that has electrical contacts suitable for insertion of a ROM.
slot	Refers to a ROM package's physical location within the sockets in a Compass or GRiDCase. A slot will occupy more than one socket if multiple ROMs are used to comprise a package.
termulator	A terminal emulator used to transfer files via a serial port connection.
uncrimped	A ROM that has not been inserted into a carrier, and consequently still has its leads straight, ready for insertion into a socket on a printed circuit board.
under the hood	Inside the enclosure of the GRiDCase (or Compass). The location of the internal ROM board on the GRiDCase.