TEK OPERATOR'S MANUAL

Part No. 070-4346-00
Product Group 57

32K EPROM PACK

12RS11 (EMPTY)
12RS12 (BLANK)

32K EPROM Pack

FIRST PRINTING JUNE 1983

Tektronix
COMMITTED TO EXCELLENCE

TABLE OF CONTENTS

INTRODUCTION

CONTENTS

These instructions contain directions for loading a 32K EPROM pack with the contents of up to four full 8K RAM packs. (More than four RAM packs may fit in one ROM pack if they are less than full.) This allows for indelible storage of reference memories and setups. For example, this might be a very useful way for a central service organization or a manufacturing engineering group to provide 1240 reference information to their technicians in the field or on the floor.

REQUIRED EQUIPMENT

The procedure outlined in these instructions requires a 1240 Logic Analyzer, a RAM pack, a COMM pack, a host computer, an EPROM burner, and either an empty EPROM pack (12RS11) or a blank EPROM pack (12RS12). If you purchased an empty pack, you will need four Motorola 68766s or 68764s or their equivalents.

SERVICE INFORMATION

Service information for the EPROM packs is contained in the *1240 Logic Analyzer Service Manual.*

THEORY

FORMAT

The same basic format is used by both ROM and RAM packs. Each contains a header, a directory, a number of files, and a trailer. Except for the checksum in the pack trailer, all 16-bit quantities in the header and directory must be stored low-order byte first. The size of the directory and the maximum number of files may vary. Refer to Figure 1.

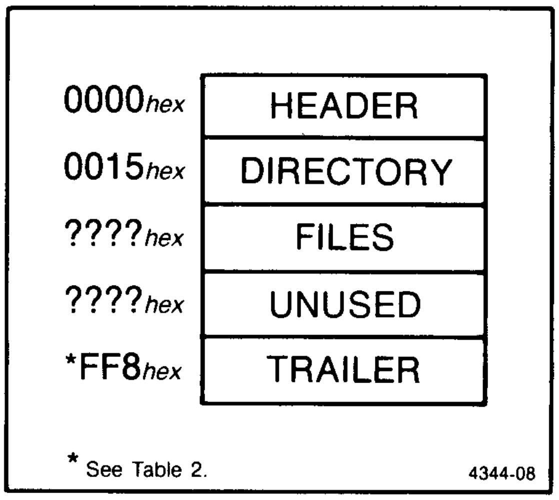

Figure 1. Basic ROM and RAM Pack format.

PACK HEADER

The first 21 bytes of the pack are reserved for the header. The header consists of:

Byte 0: "pack ID" code (01). In a user-generated pack, the "pack ID" byte must always be 01.

Bytes 1-2: length of directory (in bytes). The directory length is variable and indicates the amount of storage allocated to the directory; it does not all have to be used. This entry is stored low-order byte first. For example, a length of $40_{decimal}$ bytes should be stored as $28{,}00_{hex}$.

Bytes 3-4: pack trailer address. The pack trailer address (the eighth byte from the end of the pack) allows the 1240 to find a ROM trailer in the pack. The address of this trailer is placed in bytes 3 and 4, low-order byte first. For a 32K pack, this will be $F8{,}7F_{hex}$.

Bytes 5-20: all zeroes.

DIRECTORY FORMAT

The directory immediately follows the pack header. A directory contains a variable number of entries. There must be a directory entry for each file in the pack. The last entry in the directory must always be the UNUSED entry. It acts as the directory terminator. Only one UNUSED entry may appear in the directory.

There are five types of file entries, corresponding to the four file types and the UNUSED entry. Refer to Table 1, Directory Entry Format and the *Files* section, both following.

NOTE
All 16-bit values should be stored low-order byte first.

FILES

Files are placed immediately following the directory. They need not appear in the same order as they appear in the directory or be contiguous, but they must not overlap. The following four types of files may appear in 1200 Series ROM and RAM Packs:

- INSTRUMENT SETUP - 1240 setup
- MEMORY IMAGES - 1240 memory
- RADIX TABLE - Radix table
- RESERVED - Special-purpose internally-generated files

INSTRUMENT SETUP and RADIX TABLE files have fixed (but different) lengths. Only one RADIX TABLE file may appear in a pack. Any RADIX TABLE files after the first one are ignored.

MEMORY IMAGE and RESERVED files may vary in length. RESERVED files are files built and used by some Tektronix-generated ROM packs.

The internal formats of INSTRUMENT SETUP, MEMORY IMAGE, and RADIX TABLE files are described in the COMM pack manual.

TRAILER

The last eight bytes of a user-generated pack are reserved for the ROM trailer. In a 32K pack these are the bytes from $7FF8_{hex}$ to $7FFF_{hex}$. The trailer address in a user-generated pack depends on the capacity of the ROM pack.

Table 1
DIRECTORY ENTRY FORMAT

Byte (decimal)	Description
INSTRUMENT SETUP:	
0	type code (01)
1	length of directory entry (12)
2-3	address of file
4-5	length of file in bytes
6-11	file name (1240 display codes)
MEMORY IMAGE:	
0	type code (02)
1	length of directory entry (12)
2-3	address of file
4-5	length of file in bytes
6-11	file name (1240 display codes)
RADIX TABLE:	
0	type code (03)
1	length of directory entry (6)
2-3	address of file
4-5	length of file in bytes
RESERVED:	
0	type code (16-63)
1	length of directory entry (18)
2-3	address of file
4-5	length of file in bytes
6-11	file name (1240 display codes)
12-17	type label (1240 display codes)
UNUSED:	
0	type code (00)
1	length of directory entry (6)
2-3	address of first byte of unused space
4-5	length of unused area in bytes

Table 2
ROM PACK TRAILER ADDRESSES

ROM Pack	Trailer Address
8K	1FF8
16K	3FF8
24K	5FF8
32K	7FF8

CREATING A ROM PACK

You may place the contents of up to four RAM packs in one ROM pack. (Even more, if they are not full.) Figure 2 shows how the contents of the four RAM packs are reorganized for storage in one ROM pack.

When you put more than one RAM pack into a ROM pack, you must modify the pack header, combine the directories and the files, and compute a new checksum for the trailer.

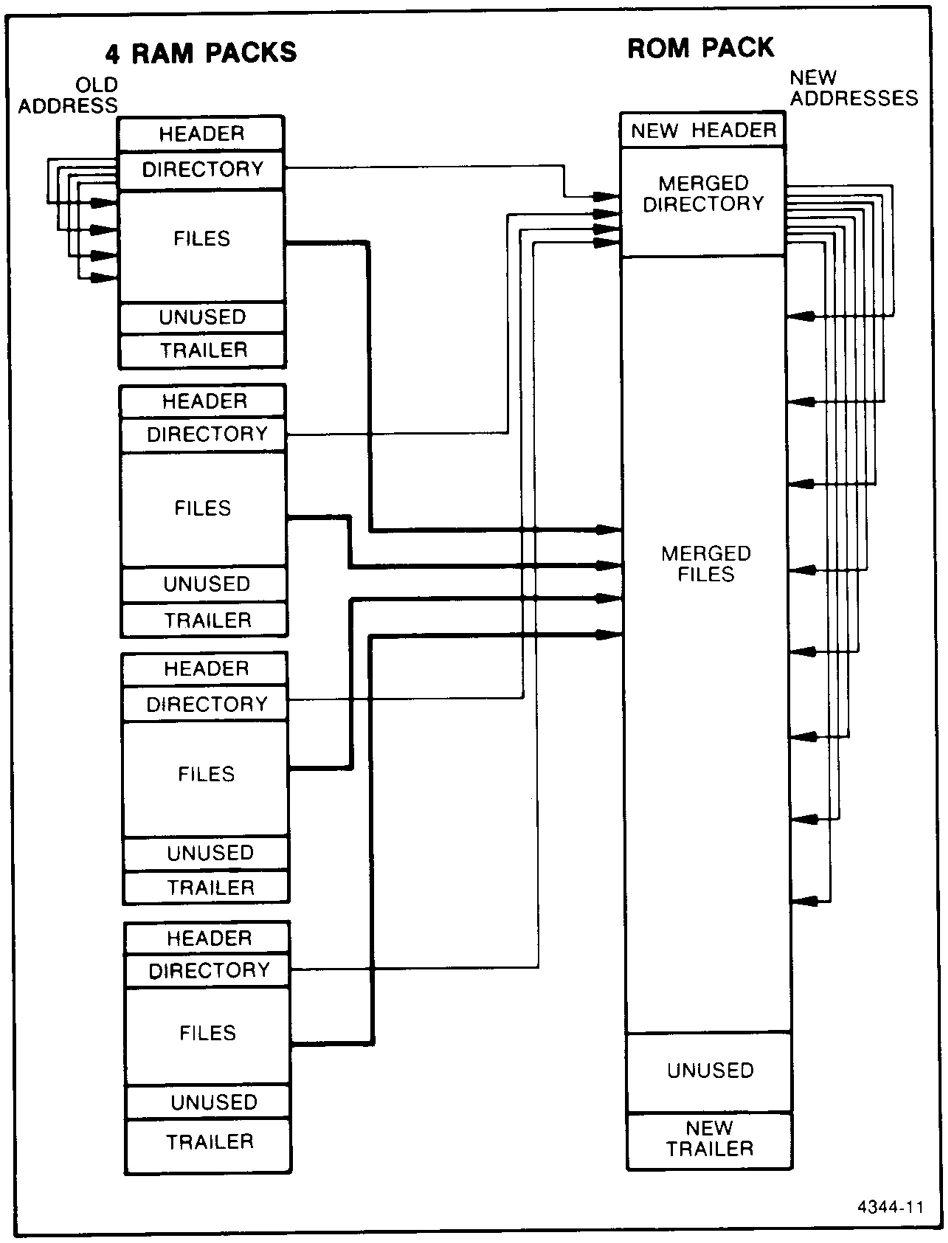

Figure 2. How up to four RAM packs are reorganized for storage in one ROM pack.

TRAILER FORMAT

In all ROM packs, the least significant bytes of the trailer addresses are F8-FF. The following describes the Trailer format.

F8 — The most significant digit in the Trailer address prefixed by a 1. See Table 2. For example, 17 for a 32K ROM Pack.

F9 — The first and second most significant digits in the Trailer address. See Table 2. For example, 7F for a 32K ROM Pack.

FA — 00

FB — 00

FC — 00

FD — FF

FE — High order byte of the checksum.

FF — Low order byte of the checksum.

UPLOAD

Upload, into the host computer, the contents of the RAM packs that you wish to put into the ROM pack. Use the instructions in your COMM pack manual.

HEADER

Place 01 in the "Pack ID" byte of the first header and delete the other headers (first 21 bytes).

DIRECTORY

Count the number of files that will be in the new ROM pack. Setup and memory directory entries each require 12 bytes, radix table and unused directory entries 6 bytes, and reserved directory entries 18 bytes. Multiplying the number of each type of file by the length of the directory entry for that type of file and summing the results will determine the size of the whole directory. Enter this directory size in bytes 1 and 2 of the HEADER, low order byte first.

Place the value of the last location in the pack, minus 7, in the pack trailer address of the HEADER, low order byte first. For example, in a 32K pack this will be 7FF8. So byte 3 will contain F8, and byte 4 will contain 7F.

FILES

Concatenate all of the files from the RAM packs and place them immediately following the directory.

CHANGE DIRECTORY

Put directory entries for each file in the one directory which will be the directory for the ROM pack.

For each directory entry, update the file address. Terminate the directory by adding an UNUSED entry.

TRAILER

Compute the new pack checksum. The checksum is a 16-bit value. To calculate the checksum for a user-generated pack, use the following procedure (shown in pseudo code):

- Initialize the variable CHECKSUM to 0.
- Set the variable PACKSIZE to the number of bytes in the pack (8000_{hex} for a 32K ROM Pack).
- Initialize the variable POINTER to the value PACKSIZE - 3 ($7FFD_{hex}$ for an 32K ROM pack).
- While POINTER$\geq$0

```
    Clear the carry flag.
    Rotate CHECKSUM left through the carry (16-bit rotate; carry = msb, lsb = 0).
    Move the byte pointed to by POINTER to the low end of TEMP and set the high-
    order byte to all 0s.
    Add-with-carry TEMP to CHECKSUM.
    POINTER = POINTER - 1
```

- Store CHECKSUM in the last two bytes of the pack (high-order byte followed by low-order byte).

To verify your checksum routine, run it on an 8K ($8192_{decimal}$) block of 55's$_{hex}$. The computed checksum should be $542B_{hex}$.

DOWNLOAD AND BURN

You should now be ready to download the ROM pack image and load your EPROMs. To gain access to the EPROMs, remove the four screws that hold the pack together. The total 32K of ROM contents must be loaded into four 8K EPROMs. The relationship between the locations of the particular EPROM's and the addresses associated with them are shown in Figure 3.

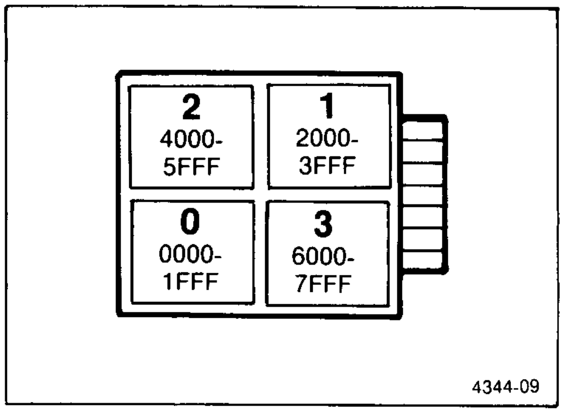

Figure 3. Relationship between physical location and address contents.

CAUTION

Observe static precautions to avoid damaging the EPROMs.