MEMORANDUM

To: Software Distribution List and NCC Personnel

From: Joel Levin

Subject: IMP Verification

Date: March 7, 1973 SD#63

With the current capabilities of the PDP-1d and the IMP system, it should be possible to verify the core image of a running IMP system against the binary files stored on the Fastrand--and in fact it is. This memo describes the four programs now written (more are coming) to enable and assist verification: IMPVER, IMGADD, IMGLIST, IMGEXPUN.

1. IMPVER

IMPVER runs by requesting from IMP DDT a dump of a portion of the IMP's core, decoding the character string returned, and comparing this with the contents of a core image created by IMGADD. It accepts any desired address limits for verification, permits optional verifying of core locations assembled as zero, and is moderately good about not looking at core which has not been assembled or patched into.

To operate: call up "IMPVER" under DDT (proper break coding to stop it outside of DDT hasn't been added). It will ask for a core image number. Answer with the number of the core image you wish to check against. If it exists the comment text (see IMGADD) will be typed out. Next it will ask for a destination (octal) IMP number. If your answer is reasonable, it will ask for lower and upper limits, and whether zeroes are to be checked. All locations in the IMP which do not agree with the image between the specified limits will be typed out in the format indicated.

The program then returns to ask for a new set of limits. Answering any question with a single altmode will halt the program. Interrupting the program with the break key is safe at any time.

2. IMGADD

This is the program which creates and patches core images used by IMPVER. The data structures of core images is given as section 5 of this memo.

IMGADD begins by asking for an image number, from 0 to 99 inclusive. If a core image does not exist for that number, one will now be created. The program requests a list of binary input files from which to create the core image (which may be TIP files as well as IMP files). The list is terminated by answering the file question with a single altmode. IMGADD now asks for descriptive text in the nature of comments. Each comment line should be terminated with an altmode, not a carriage return, and comments are terminated by a single altmode (i.e. a null line). IMGADD will prefix each comment with a slash (this is for some future implementation of something that will suck up its own file list ignoring the comments). The list of file names and comments is written as a descriptor item associated with that core image; the comment portion is typed out by IMPVER when the core image is specified. After inputting comment text, IMGADD proceeds to build the core image, processing each file in the list sequentially (patch files hence must be specified in the correct loading order).

If the core image given to IMGADD is one which already exists, IMGADD asks if you with to add a patch file (or more) to the image. If you made a mistake, you may abort by replying "N"; if you are adding patched to the image, answer "Y." IMGADD will type out the current file list and then ask for file names as above. When the input list is terminated IMGADD will type out all the existing comment text, then pause for more. Following input of a null line the patch files will be added to the core image in the usual manner.

3. IMGLIST

This is a program which lists all the currently existing core images, in either a short form or a long form. When you start IMGLIST and specify the form, either the comment text or the entire descriptor created by IMGADD will be typed for each image stored on the drum.

4. IMGEXPUN

When a core image is obsolete, this program can erase it. It asks for a core image, types the entire descriptor item, then asks "ARE YOU SURE..." If the answer is not the full word "YES" the program halts; otherwise the core image is expunged.

5. CORE IMAGE STRUCTURE

All data is on third 2 of the Fastrand, where network IO is done.

The master Image TOC is an item pointed to by an invariant number. The length is 102. words consisting of the usual two words of overhead and 100. drum pointers to image TOCs. A non-existent image is indicated by a pointer of -0.

An image TOC is 70. words long: six of overhead and 64. pointers to core page images. Pages which have nothing assembled have pointers of -0. Image TOC overhead consists of the usual two words, then the DRA of the descriptor item, the initials of the saver of the core image, and two words of time and date in standard PDP-1 form.

Each 512. word page of X16 core image has an associated item (if there is any assembled code on that page). This item is 514. words long, consisting of two words of overhead and 512. words of X16 binary. At present each word containing a word of X16 code has bits 0 and 1 cleared. Words on the page which have not been assembled contain -0.

Each image has associated with it a text item containing the file list and comments. This item is of variable length. It is in TOS format--the entire character string is terminated with a single EOM, with a double EOL (76) separating the file list from the comment text.