FILE: HANDBOOK

Compliments of FRESS

A File Retrieval and Editing SyStem

Release 9.1 2 MAY 79

FRESS staff 'Handbook'

Compiled by Alan Hecht, summer '78

(to be digested with grains of salt)

1 FILE HANDLING

PAGING

- 1600 Byte pages
- FRESS does it's own paging
- Arbitrary number of pages in a file
- Doubly linked list of pages (in and out of core) (see figure 1)
- Hold page 0 and whatever needed to do current task, i.e. pages on which LP's are being resolved.
- Garbage collection -- from free page list
- Holding pages and modifying them -- flags are set
- Spill file for 'revert' -- save old versions of all affected pages. To revert means to reset pointers of adjacent pages to old pages
- There is no protection against a crash during a write.
- Finding a free page slot

    1) Free page from free page pool pointed at in UCB (User Control Block)
    2) Unheld, unaltered page in current file (file's free list)
    3) Unheld, unaltered page in another file
    4) Unheld, altered page in another file
    5) Abend

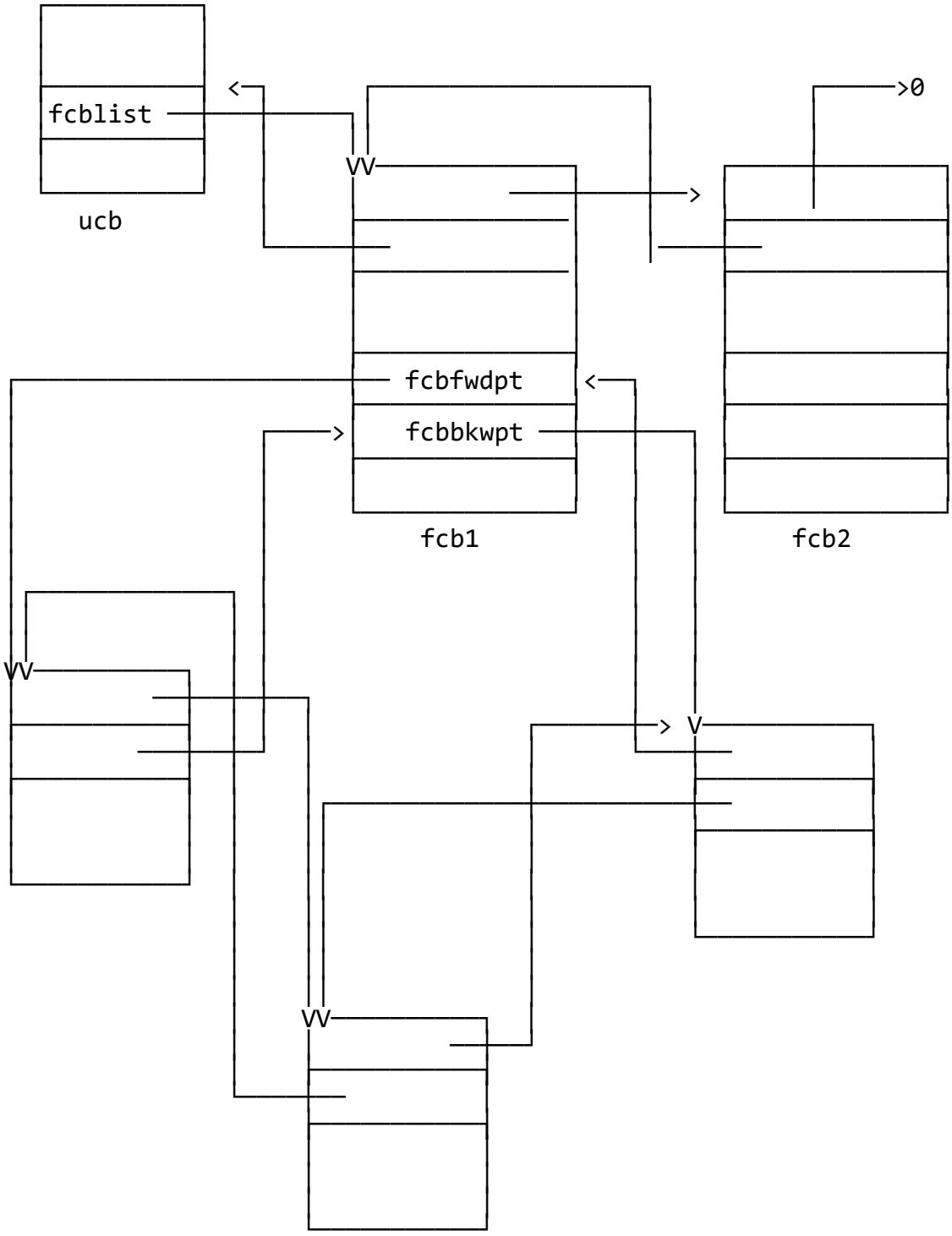


Figure 1: In Core Page Connections (Similar Rings Exist for All Spaces)

2 FILE INTERNALS

FRESS PAGE

- 1600 Bytes/Record = 1 FRESS page (see Figure 2)
- The first few bytes of a page have pertinent information (see figure 3)
- When you create a file you get:

  Page 0 -- Page Control Block (PCB) -- ONLY ONE per file (see Figure 4)

  Page 1 -- Password page

  Page 2 -- Internal Label space (always at least two - for top and bottom of space) (see Figure 5)

  Page 3 -- Structure Space (see Figure 6)

  Page 4 -- First page of text space

  Page 5 -- (not used right away)

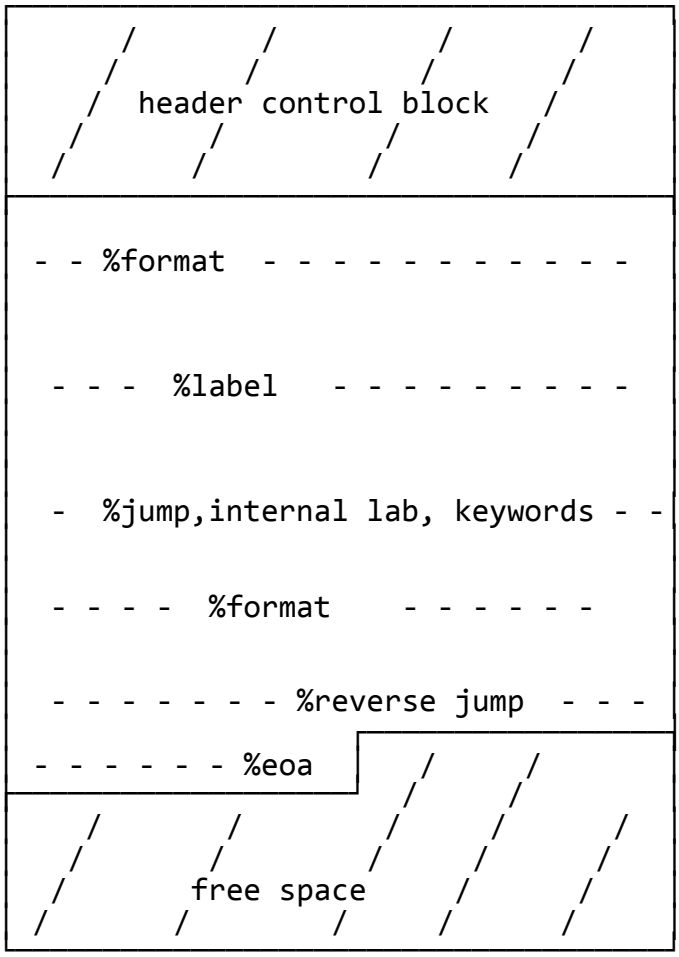


Figure 2: Normal FRESS page

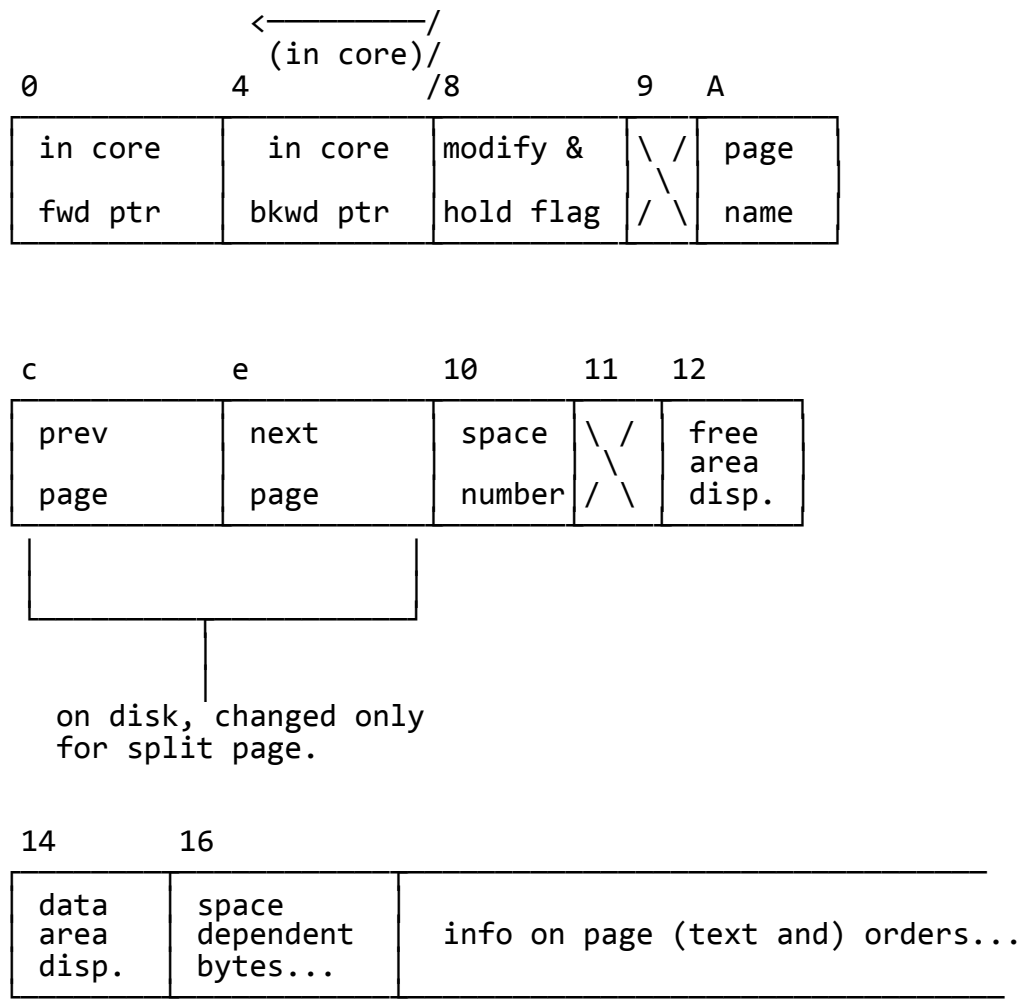


Figure 3: Header Control Block for Normal FRESS Page

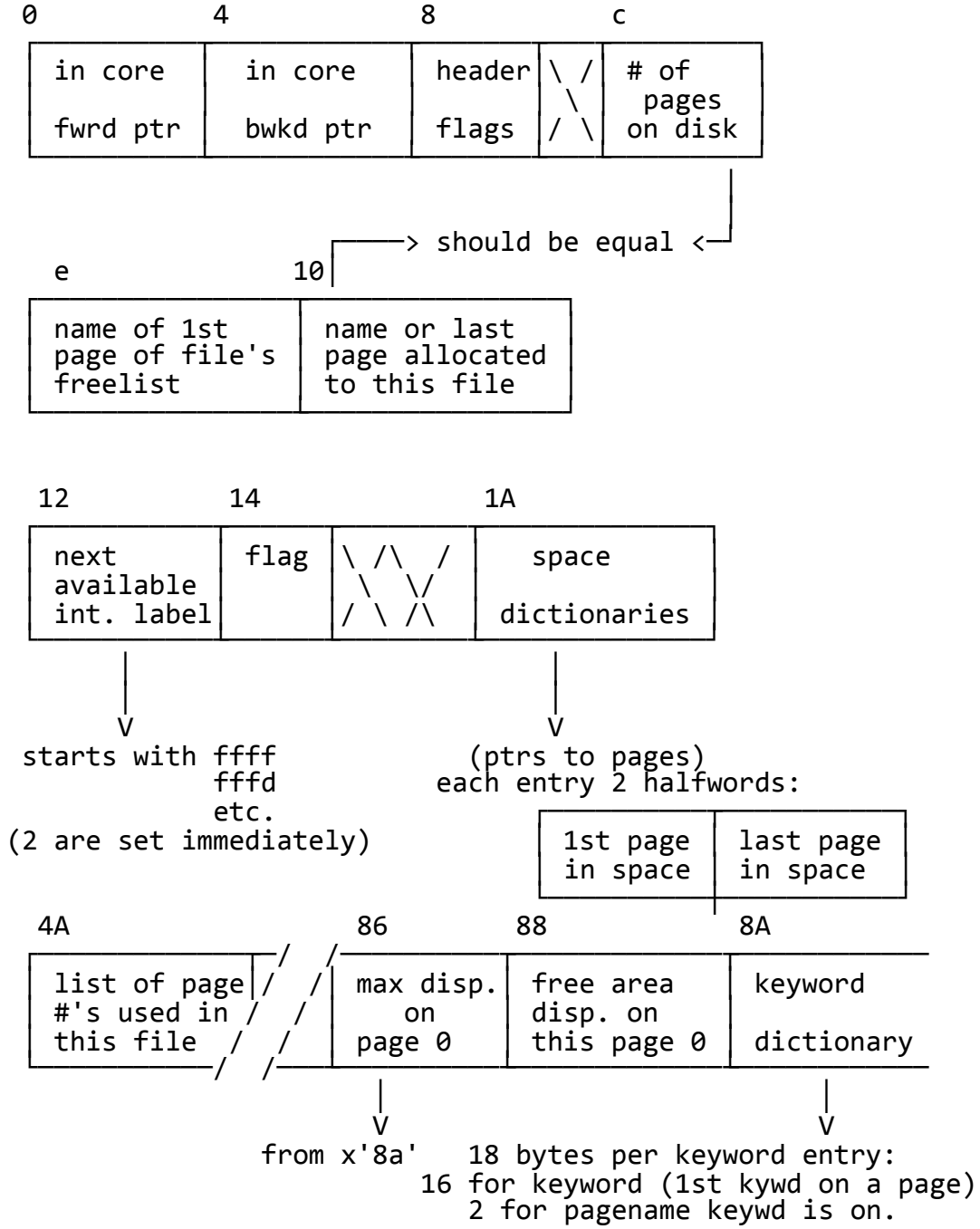


Figure 4: Page 0 Header Control Block

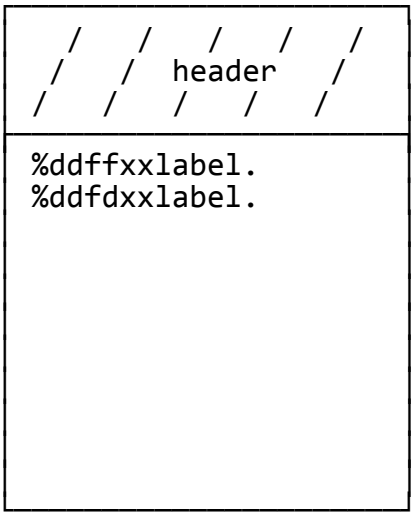
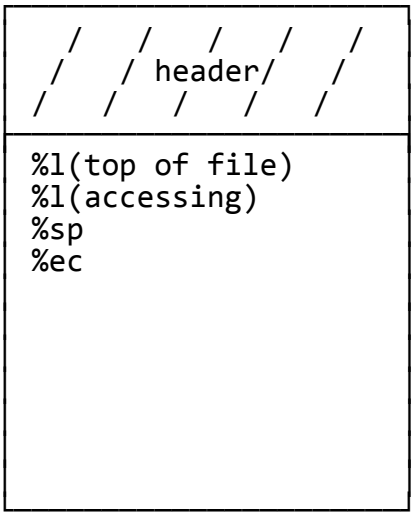


Figure 5: Label Display Space



ex.

%op mod intlab info 00

n.b. The structure space is a hypertext only extract of the text space, and the label display space is a label only extract of the structure space (so lookup there is very fast).

Figure 6: Structure Space

3 DATA STRUCTURE

TEXT STRINGS, FORMAT CODES, AND STRUCTURE CODES

- Structure (in a text space) can be:
  1) keywords
  2) internal labels
  3) (text could be related to structure)
- Structure Orders
  - Format

    x'6c'<opcode ><modifier><internal label> (<data fields>)
    ex: % < 80 FFXX
    1 Byte 1 Byte 1 Byte 2 Bytes
- Possible Opcodes
  1) Points - location to anchor label
  2) Tags - (Area Orders, Annotation Tags, Decimal Label Reference Tags, Picture Reference Tags)
  3) Block Start and End (annotation blocks, decimal label blocks)
  4) Table Orders - Only in label display and keyword display spaces
  5) Jumps and Pmujs (Splices and Ecilps)
  6) End of page - where free area starts
- Spaces (see Figure 7)

  Protect Pages (for passwords)

  Main Text Space

  Annotation Space

  Work Space

Structure Space

Label Display Space

(note: editing in the structure or label display spaces will be reflected in the main text space, annotation space, and in the work space)

Internal Label Space

Dslink Space -- Data Structure (interfile structure)

Picture Space

- Control Blocks
  - Storage Management

    UCB - (User Control Block) (not file specific) status of user per session (ptr to FCB; global msgs, etc.); global status info; dynamic storage management of other blocks

  - File Handling and Paging

    FCB - (File Control Block (file specific) global info (one for every file) (PCB part of this).

  - Displaying a file

    WCB - info about window and text in it

    Viewspecs - how file displayed

    SCB - stack control block - manages others

    JRS - jump return stack

    MRR - memory return ring

    BTL - Block trail (discrete and continuous)

    SRS - Splice return stack -- to scroll backwards through splices

    IRS - instance return stack -- for tags to annotation blocks

    SCOPE - info about a hit point

qualifiers -l label

-w word

-b block

-o order

-c character

- Command Language Interpretation

  Data Structure Pointer (DSPTR) - info about a hit point

  Function Area - Status of system with respect to commands

  Fuction Element - single command info.

  Function Table - info about each function's syntax

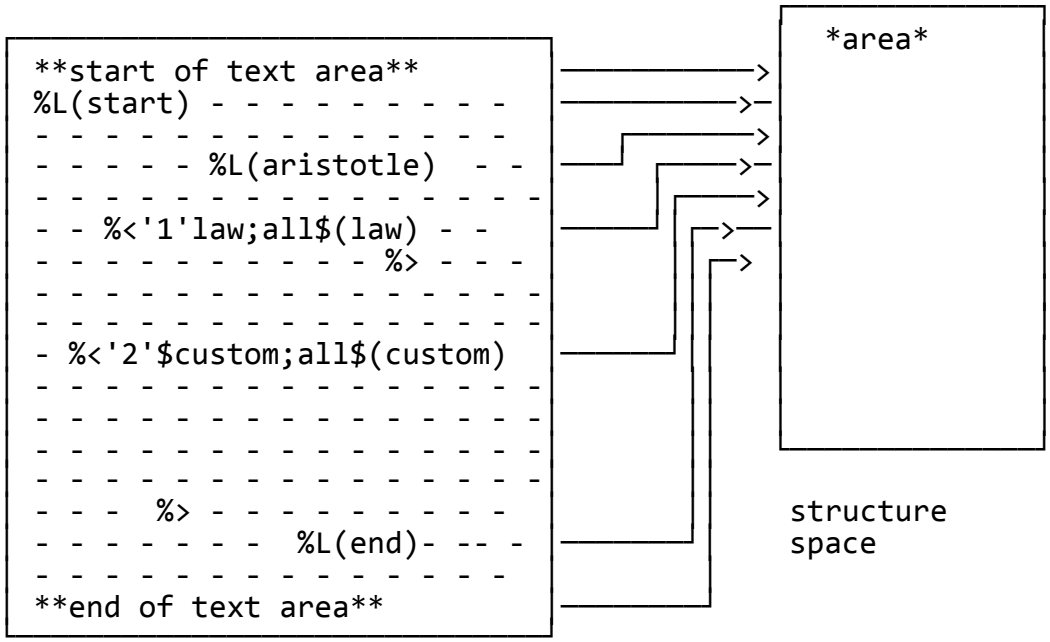


Figure 7: Typical File - Spaces

CREATION OF STRUCTURE

Made 'on the fly' -- creates structure orders as needed, pushing text down on the page.

DISPLAYING STRUCTURE

Also 'on the fly' -- special cases as display goes through text space, keeping in mind viewing specifications.

ACCESSING STRUCTURE

Through STRUCTURE space: in structure space is a structure order for every piece of structure, stored in order that they are in the file. With every piece of structure is associated an internal label. The internal label space is referenced, the page where structure exists is found in the internal label space; the page is then referenced. (see Figure 8)

ex: For a Get Label command (GL), FRESS does a search through the label display space, finds the associated internal label, then does steps (2) and (3) of figure 8.

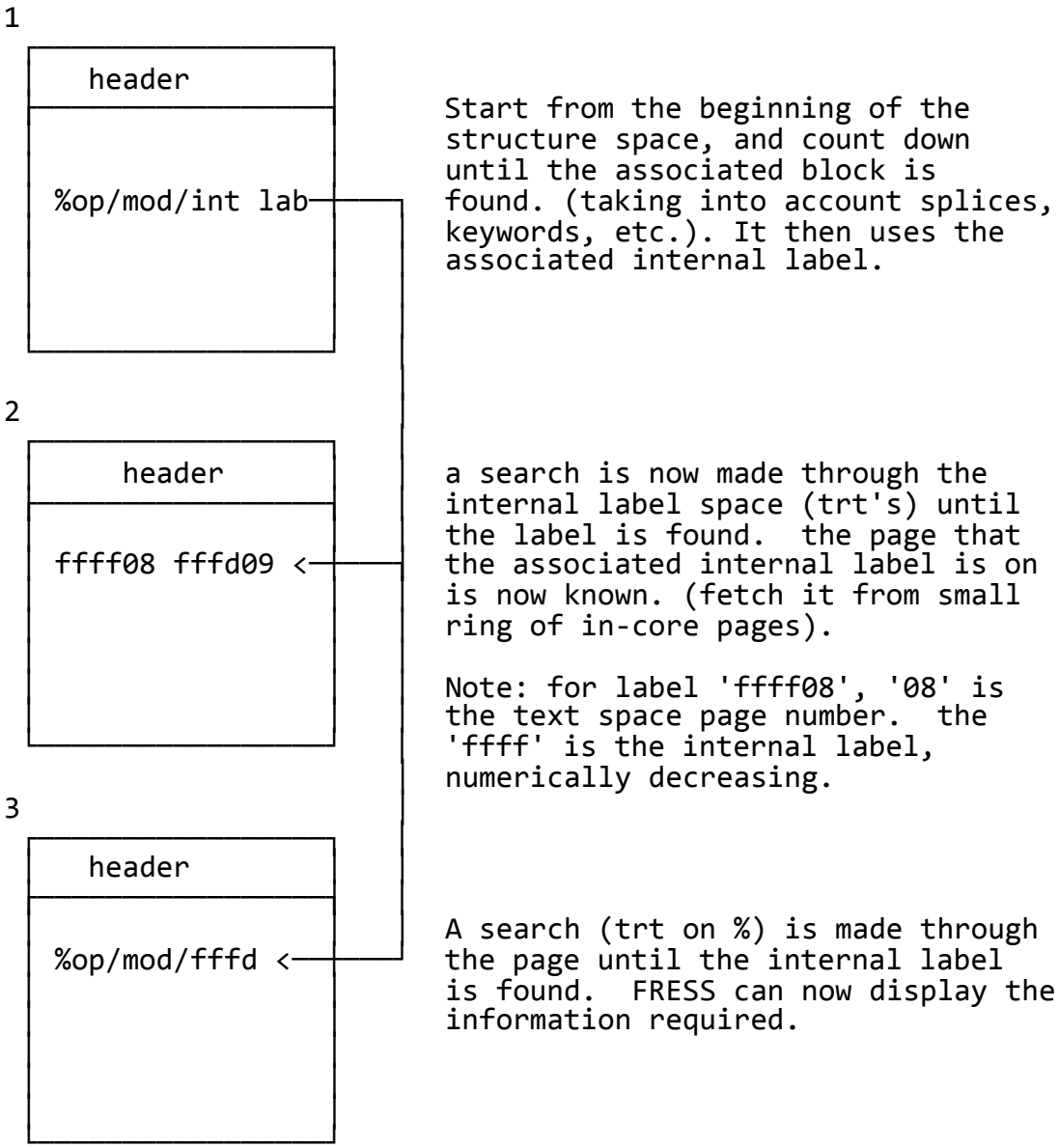


Figure 8: WHAT FRESS does on a get decimal label command (gdl)

Note: Sometimes a user will get a message that a keyword is too long when it really is not. This is due to the fact that when editing changes in certain types of areas are done with "inserts" and "deletes" (see SUBSTITU command), the "insert" is executed before the "delete", and thus may cause the keyword to exceed its limit.

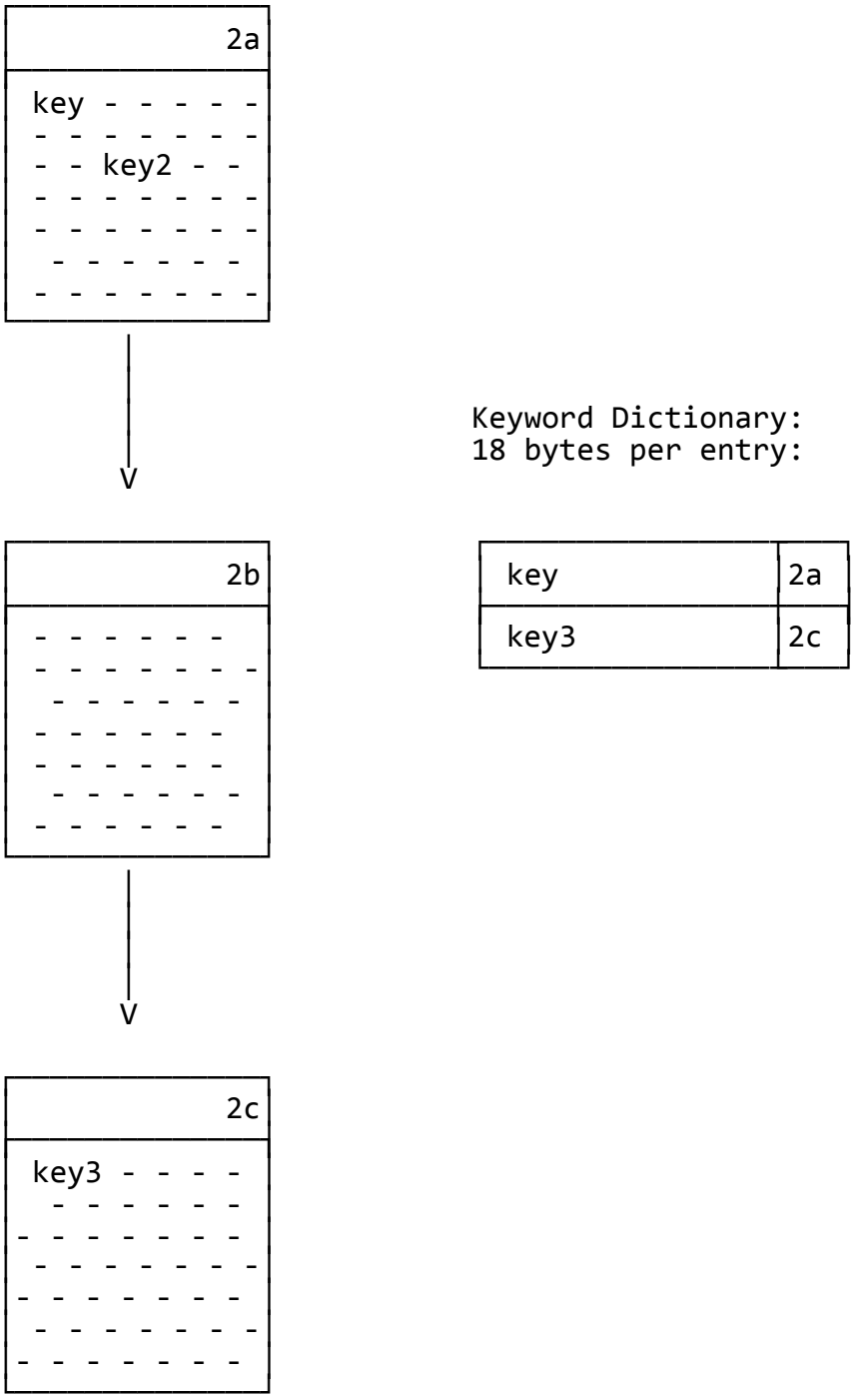


Figure 9: Keyword Dictionary (contains only entries that begin a page in keyword space)

CLINTERP

- Searches forward for CR and illegal characters in command line, then back for dot convention.

- Analyzes each Function and parms

  Sets function element (FUNCELEM) in Function Area (FUNAREA)

  Sets up a translate table for first character of function (ex: '&&&' = house function, '%' = hypertext function)

- Normal Commands:

  Sets up codes in FUNAREA

  Parses parms (required parms pointed to first, then optional left to right).

  Sets opcode (FUNCOPCD) and flag (FUNCFLAG)

  Sets parms in FUNAREA (ptrs).

6 TAPE

READING TAPE

To access tape from which to get a file:

1) MOUNT TAPE <slot number> PASS <password>

Each type of tape has a different slot number (bug files=a317ah,fress crappola=a314fd,demos=c643fr,and source code=a645fr),but the password on all the tapes is chinainn (chosen because of a past tradition of frequenting a Chinese restaurant of the said name). All pertinent information about the specific tapes is located in the FRESS maintenance notebook.

To insure that you do not cause the tape to run off its spool:

2) TAPE REWIND

To get to the section where the file is located:

3) TAPE FSF <tape mark>

To put file onto disk (automatically goes to A-disk):

4) TAPE LOAD <filename> <filetype>

To finish session with the tape:

5) TAPE REWIND
6) DET <address where attached>

TO WRITE TO TAPE

If one wants a tape in write mode, one must show the operator his ownership card (in FRESS folder) and bring its ring (located in the top draw of the FRESS desk).

A)First one has to put all the files one wants together within two tape marks into an EXEC. The easiest way to group the files together is to put a similar prefix onto all of them. For example, bug files all begin with the prefix "@". Then using the LISTFILE command, create an EXEC with all those files beginning with "@".

LISTF * FRESS * (EXEC L

1) LISTF <filename> <filetype> <disk> (EXEC
<attribute option>

Then rename the CMS EXEC file which is created by the LISTFILE command to whatever you want. For instance, a bug file would be renamed to the next consecutive number up of those bug files already created.

RE CMS EXEC a1 bug9 = =

2) RE <old filename> <filetype> <disk> <new filename> <filetype> <disk>

B) Now to write the EXEC onto tape:

1) MOUNT TAPE <slot number> PASS <password>
2) TAPE REWIND

The next thing to do is to get to the point after where the last files present on the tape are located. This is done by a TAPE SCAN (which prints out all the files between two tape marks), or TAPE FSF (which takes you some particular number of tape marks forward). The important consideration is not to read past the EOF tape mark and have the tape physically run off its spool. The tape marks are the only things which keep the tape under control. In the case of writing onto tape,this possibility becomes more probable since one must get to the end of the tape so as not to write over any material that is already stored there.

3) TAPE FSF <tape mark>

To put the EXEC onto tape:

4)TAPE DUMP <filename> EXEC
5) EXEC <filename> TAPE DUMP

Then write some tape marks onto the end of the tape to help prevent reading off the tape:

6) TAPE WTM <number of tapemarks>

To finish session with the tape:

7) TAPE REWIND
8) DET <address where attached>

GENERAL INFORMATION ON TAPES

Always remember to rewind the tape before detaching.
All tape commands begin with the word TAPE.
Make sure you do not write over any files already present on the tape.
Make sure you do not read off the end of the tape.
Files on the source tape (a645fr) which are of the type +SYSIN must be expanded to use, while those of the type SYSIN need only be renamed to ASSEMBLE.

DEBUGGING FRESS FILES

There are 5 spaces in a FRESS file for which routines have been written to fix errors. They are: text(t), structure(s), internal label(i), annotation(a), and work(w).

To use the FRESS debugging commands, one must change the password to the account either when one accesses the file or once one is within the editing mode of the file.

A)To access the file with a password:
1) GF <filename> STAFFDBG

Once having changed the password in this way, one can check all the spaces at once with the command:
2) CK

--See section C for the correction of bugs--

B) To access debugging commands after entering the file:
1) CP SYSDBUG
This now allows checking of each space individually.

To check a specific space:
2) CS <space abbreviation>

C) Checking a file in the way outlined by section A or B will yield information about the internal state of the file. If the STAFFDBG password is used and the message FILE OKAY is returned, then the file is fine as far as the FRESS debugging commands can tell. If the SYSDBUG password is employed and the message SPACE OKAY is returned, then the specific space is determined to be all right.

If the debugging commands detect an internal error, however, messages will be returned describing the character of the error. These error messages and the ways to repair the files are described in the DEBUGGING section of the maintenance notebook.

An example of a common error which occurs and the way to fix it is the occurrence of unprintable characters within the text space.
CHAR <page number> <offset>
means a bad character exists at th offset specified on the page number specified. To correct this error so that the user can work within his file and change those c characters to the ones intended, patch in blanks where the bad characters occur.

To get the page where the bad character is:

1) GP <page number>

Display the offset where the mistake supposedly occurs to make sure it is a bad character.

2) DO <offset>

To patch in the EBCDIC representation of the blank:

3) PA <offset> 40

Check to make sure that you patched in the blank at the right address:

4) DO <offset>

After all of the bad characters are fixed, check the space again to make sure that all the bad characters have been fixed and check all the other spaces to make sure that no problems have been created within the file as a result of the changes.

5) CS <space abbreviation>

TO DEBUG A FILE

(a)Get all the information from the FRSCOMM

(b)Execute the instruction the user specifics in the FRSCOMM to make sure the file really does not work.

(c)Debug the file with the FRESS debugging commands.

(d)Execute the command that the file initially blew up on to make sure that it can now be done with the change made.

(e)Send the file back to the user by linking to their account, but before sending it back call the user and inform him so that they will know that their file is fixed and so that you do not destroy any changes that the user may have made in his file since sending it to you. Make sure changes made on a person's blown file are made permanent by doing an ACCEPT at the end of the session:

6) ACCEPT p 1

GENERAL INFORMATION ON DEBUGGING:

Set the number of lines to be printed to at least three with the SET DISPLAY command when debugging because the smaller number of lines one views, the more chance there is that the space will

have to be rechecked several times before finding all the errors. Write down all the errors FRESS finds to keep an accurate account of the steps followed to repair the file and save money from unnecessarily employing the check commands.
After making changes on a person's blown file, make sure you do an ACCEPT. Otherwise, at the end of the session, you will lose the changes that you have made.

BOOTSTRAPPING THE SYSTEM

If the cross loader is not in core, read in the paper tape. Set the TTY-ADC switch to TTY. Turn the teletype to LINE (if the teletype chugs, set the SINGLE CYCLE switch (on the left side of the control panel) and flip the START switch, then clear the SINGLE CYCLE switch. Next execute a program that will set the baud rate to 110 for the teletype--set the MEMORY PROTECT switch UP, then set the address switches to 40 and flip start. Now execute the papertape bootstrap loader --clear the MEMORY PROTECT sw(put it down), load the paper tape onto the teletype and start the reader. Leave the address at 40, put all of the top row down, and press the readnext button to see if the program to set the baud rate is there. Then flip the START switch on the IMLAC and wait until the run light goes out (All four rows of lights should blink at least at first. Some of the lights do not work on the board, however, and we do not know which ones.). Read the tape in. Set the TTY-ADC switch back to ADC.

PATCH in the addresses that are incorrect on the current paper tape.

If the TTY simulator is not in core, set 27300 (the starting address of the cross loader and the one pointed to by arrows in the address switches), and leave all the data switches (top row) down. Flip START and wait until the run light goes out, during which time the word, LOAD, should blip on and off the screen as the program loads.

Now put the starting address at 100 and hit STOP/START to logon.

SETTING THE BAUD RATE

If the MEMORY PROTECT SWITCH is down, locations 40 to 77 are read from ROM. Here lies the bootstrap loader for reading binary data on paper tape from the TTY.

If the MEMORY PROTECT SWITCH is up, locations 40 to 77 are real core. Starting at loc 40 is a program to set the BAUD rate to 110 for the TTY.

```
000040    060047              LAC DATA1
000041    001261              IOT 261
000042    001262              IOT 262
000043    060050              LAC DATA2
000044    001164              IOT 164
000045    100001              CLA
000046    000000              HLT
000047    123471    DATA1     123471 *
000050    140341    DATA2     140341
     * for 300 Baud this would be 157564, 040341
       for 1200 Baud 173734, 040341
```

ALWAYS leave MEMORY PROTECT SWITCH in the DOWN position when not in use since that protects the memory locations 40 to 77 from being overwritten accidentally.

UPDATES TO FRESS

TESTING NEW ROUTINES

When testing out a new routine, use the experimental FRESSX rather than FRESS itself so that untried changes do not disturb the entire system. Get the associated txtlibs and maclibs of the routine(s) being changed off the source tape. in

1) TXTLIB GEN NEWONE <fn,fn...>

newone specifies the name of the textlib being created. The textlib generated appears on the A

2) LOAD <fn,fn,...>

3) GENFRESSX <fn,fn...>

This EXEC generates the FRESSX module for testing routines (see maintenance notebook for contents of the exec).

TO PUT IN NEW VERSION OF FRESS

1) GETDISK

2) <Get the routine and its associated txtlibs and maclibs off oftape>

3) RE FRESSX MODULE A1 FRESS = =

4) FR* MODULE * (D

5) ERASE FRESS MODULE F

Must erase first because disk is full.

6) QCOPY FRESSX MODULE A1 FRESS = F1

7) L * MODULE F (D

Just checking to make sure it is there.

Now change the version which appears on the FRESS logon to express the difference in FRESS.

8) <Write the new routines onto the end of the source tape.>

TO UPDATE MESSAGES

1) GETDISK
2) FMACLIST FMAC
3) QCOPY FMAC MACLIB F1 = = A1
4) FMACSPLIT FMAC INFO
5) E INFO MEMO

Change the message that is written between the &C'_____', and then FILE the memo.

6) FMACREP FMAC INFO
7) QCOPY FMAC MACLIB A1 = = F1 (ERASE

MISCELLANEOUS

Internal labels go down by two's--FFFF -> FFFD.

The R/W password on all FRESS related accounts is AVD (Andy Van Dam).

Sometimes when people are having problems because of footnot if they reformat their file a little by adding or subtracting lines from places, the file may fix itself.

In an addressing exception, the error is usually at the instruction before the one specified in the message because the sy increments by one before exiting.

Any character below the EBCDIC representation of a blank ('40) is unprintable.

For problems, Steve Feiner knows a lot about the user aspects of FRESS. If there's really an emergency, Joe Stranberg is the world's most experienced FRESS user.

TABLE OF CONTENTS