Highway Accident System

System Manual

January 2005

Produced by Matthew Nicoll
Cypher Consulting
menicoll@CypherConsulting.com
250-337-8441

P:\HQ\Eng\safety\has\doc\system\HASsystem.doc
(where P: is mapped to \\Olive\S3018)

Document Revisions:

2000-12-21	Convert from WordPerfect 5.1 to MSWord97.
2001-06-14	Convert from MSWord97 Master Document to MSWord97 long document.
2001-10-17	Completed revision.
2001-11-13	Minor corrections, "Mirror Margins" set to make sections start on absolute odd pages.
2002-03-12	Fixed column numbers in THASP.LANDMARK.MATCH.LIST section. Highway Classification expansion. SWAR enhancements
2002-04-12	User Manual to PDF section updated.
2002-06-06	Revised User Setup (SPFAUTO etc)
2002-07-04	Changed Courier to Courier New in some charts so arrowheads print.
2003-01-14	Updated fatal processing instructions & chart.
2003-02-24	Added Section 11 HASLKI Maintenance, Implementing a new LKI.
2003-05-07	replaced FS_Public@HQA@TH with P:\HQ
2003-07-25	added notes on LKI file versions and proc THAS030.
2003-10-20	updated Update section for new fatal handling utilities, LOCTEXT files, new job U38, CASELIST, mods to PL/I programs 240 and 251
2003-11-03	more update section revisions
2004-04-14	updated for modified MVB and accident records.
2004-05-30	added notes on traffic volume interpolation and extending.
2004-06-01	added (sort by case) to Job U01 chart.
2004-06-10	update job U40 revised.
2004-06-16	updated ISB new user setup procedure
2004-09-28	HASutil setup/update info added.
2004-10-30	LKI match file & procedures updates
2004-12-04	revised update startup: MVB file now allocated and permitted automatically.
2005-01-16	small revisions in LKI promotion section.
2005-01-23	revised definition of THASP.SEGMENT file (added create and road add dates)

1 Introduction

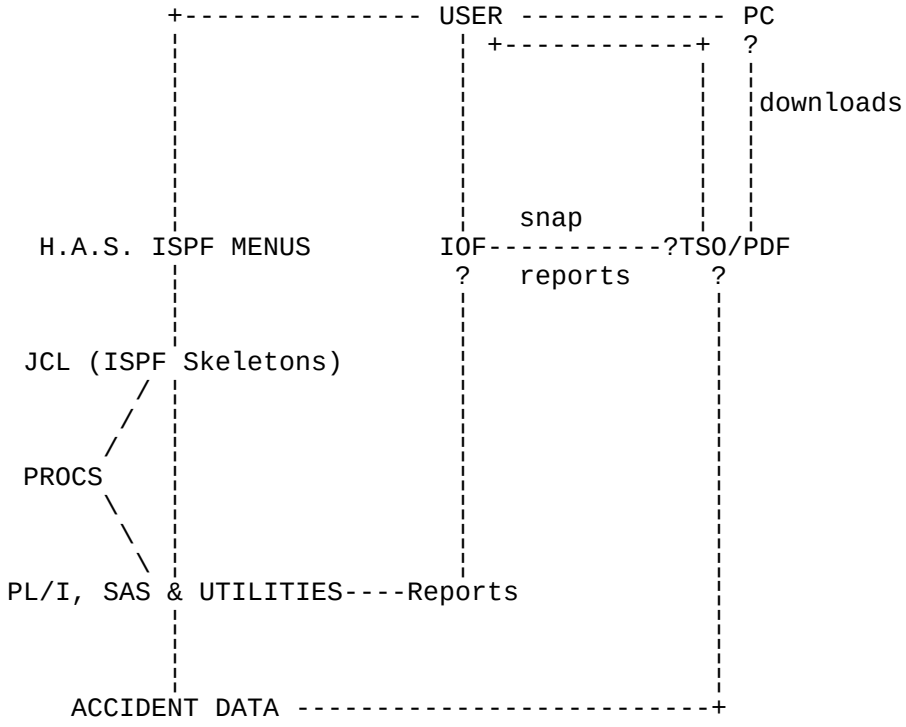
The Highway Accident System (H.A.S.) was developed in 1988 and 1989 for the Highway Safety Branch of the Ministry of Transportation and Highways, as a replacement for the Highway Safety Improvement System. The system has been used, maintained and enhanced continually since then.

The system consists of the Update, Data Retrieval, and Utility sub-systems.

The Update sub-system is used to prepare accident data obtained from the Motor Vehicles Branch for inclusion in the H.A.S. The Data Retrieval sub-system consists of accident data selection, reporting and analysis programs. The ISPF user interface allows users to integrate their own SAS programs into data retrieval batch jobs.

The programs are written in PL/I. The user interface was originally written in Wylbur, but was converted to REXX/TSO/ISPF in June 1991.

The following diagram illustrates the relationship between the user, the various software components, and the accident data:



2 Documentation

2.1 Introduction

The documentation for the Highway Accident System consists of two manuals - the System Manual, and the User's Manual. Both manuals are in MS-Word 97 format. There is an Adobe Acrobat version of the users manual.

The manuals are located in the following public locations:

```
P:\HQ\Eng\safety\has\doc\user\HASuser.doc
P:\HQ\Eng\safety\has\doc\user\HASuser.pdf

P:\HQ\Eng\safety\has\doc\system\HASsystem.doc
```

2.2 Conversion of the User's Manual to Adobe Acrobat Format

The conversion of the User's Manual into PDF format was done so that it could be reliably and consistently viewed and printed from any workstation without the font and template problems of MS-Word.

The conversion is done using Adobe Acrobat 4.0:

- Open the document in Word,
- select **Create Acrobat PDF...** from the File menu, (Installing Adobe Acrobat adds this menu item to Word.)
- Select **Use Acrobat PDFWriter**
- Press **Create**

Initially, the Select Acrobat Distiller option, with the "Print via Distiller's printer" and "PressOptimized" options was used, in an attempt to pursuade Acrobat to translate the old DOS drawing characters. This proved unreliable. The Distiller also creates links on table of contents items and cross-reference items. These also proved unreliable: these links would often "miss", and spurious links were created, in random places, linking to other random places!

2.3 System Manual Organization

Warning: Do not be tempted to place this document in Master Document Format!

Due to the (massive) problems we encountered with Word's Master Document "feature", the HAS System Manual is set up as a single long MSWord97 document, divided into Sections.

Refer to **:** http://www.mvps.org/word/FAQs/General/RecoverMasterDocs.htm for information on the horrors MS-Word Master / Sub Documents.

Each level 1 heading is preceded with a Section Break (Odd Page).

To change a section break (eg from Continuous to Odd Page):

- put the cursor in the section (ie after the section break)
- select File/PageSetup/Layout
- change the "Section Start"

Make sure that the Header at the start of each level 1 section is NOT "same as previous".

If *Same As Previous* is displayed at the upper right of the header, remove it by clicking on the Same-As-Previous toggle icon on the Header and Footer toolbar.

Note: Gremlins in MSWord like to change these periodically to Same As Previous. If the Headers are not performing properly check here first. Also, after adding or deleting any Section Breaks the Headers will alter because the Header and Footer information for the preceding section is stored in the following Section Break.

Page numbers start at 1 at the beginning of each heading level 1 section.

The section number precedes the page number in the Header, and in the Table of Contents (eg 3-1, 3-2 etc.). This is achieved as follows, using the floating Header and Footer toolbar which pops up when you select View/Header and Footer:

- Insert Page Number into the Header of the Section.
- Select: Format the Page Number
- Select: Include Chapter Number
  Heading 1
  Start at Page 1 for numbering

To ensure Odd Page Breaks really work:

- The "Odd Page Section Break" works on the "given" not the absolute page number, so it does not by itself cause the next printed page to be on an "absolute" odd page number.
- To ensure that a section starts on an absolute odd page (i.e. a new piece of paper when duplex printing), go to File/Page Setup, select "apply to whole document", then check Mirror Margins.
- This will also cause "next page" section breaks to be absolute "odd page" section breaks!
- So if you want a section+page break which can go to an absolute even page number, you have to insert a continuous section break, then a page break!

Place the heading 1 text as a Cross Reference in the Header:

Place the cursor just inside the section (eg on the level 1 heading)
View / Headers and Footers
Insert / Cross Reference in the centre tab of Header

- Choose: Reference Type: Heading
- Reference To: Heading Text
- For which Heading: Choose the corresponding Heading for the Section from the drop down list.

2.4 HASINIT Style

The HASINIT Style originated in the WordPerfect documentation and is still retained in parts of the documentation, especially where there are many tabs set, and where vertical spacing is important.

The line spacing in HASINIT is exactly 10.8 pts.

HASINIT Style:
Font:Courier, 10 pt.(FE)English(US), (Other)English(US),
Char Scale 100%, Flush Left, Line Spacing Exactly 10.8 pt.
Don't hyphenate, Body text, Tabs: -1", -0.5"

3 Development Environment

3.1 Libraries

The following Development libraries are used for doing development on the Highway Accident System.

```
THASD.SRCELIB          - PLI source
                       - member names THASnnn

THASD.OBJLIB           - PLI object modules
                       - technically, these are non-executable load
                         modules

THASD.LISTLIB          - PLI compile listings

THASD.LOADLIB          - PLI load modules (executable)

THASD.TEXTLIB          - PLI 'include' code
                       - named THASxaaa, where 'x' is:
                         - R - record descriptions
                         - O - overlays of record descriptions
                         - F - fields of records
                         - X - external data declarations
                         - D - other data (e.g., variable lists)
                       - currently, overlays are named THASRaaO
                         instead
                         of THASOaaa as was originally intended
                       - Note: this scheme has not been strictly
                         adhered to, other than X for external.

THASD.CARDLIB          - small control files
                       - named THASxaaa, where 'x' is:
                         - S - Sort control cards
                         - M - Merge control cards
                         - T - Tables

THASD.PROCLIB          - Catalogued Procedures

THASD.JCLIB            - JCL files
                       - &parm. is used for strings which are to be
                         replaced by a REXX EXEC.
                       - includes JCL to run PL/I compiles and links

THASD.SASLIB           - SAS programs

THASD.TESTJCL          - JCL for testing programs etc.

THASD.UTILJCL          - JCL for running utility programs
                       - file RUN9xx runs THAS9xx in THASD.SRCELIB

THASD.ISPF.ISPCLIB     - REXX programs part of the ISPF Dialog

THASD.ISPF.ISPPLIB     - ISPF Panel definitions

THASD.ISPF.ISPSLIB     - JCL Skeleton files

THASD.ISPF.ISPMLIB     - ISPF Messages

THASD.REXX.EXEC        - REXX programs which are not part of the ISPF
                         Dialog (utilities, startup etc.)
```

Following are the corresponding (and other) Production libraries. When a development project is complete and tested it is promoted from the Development (THASD) libraries to the corresponding Production (THASP) libraries.

```
      THASP.SRCELIB
      THASP.OBJLIB
      THASP.LISTLIB           - for compiles done in production libs
      THASP.LOADLIB
      THASP.TEXTLIB
      THASP.CARDLIB
      THASP.PROCLIB
      THASP.ISPF.ISPCLIB
      THASP.ISPF.ISPPLIB
      THASP.ISPF.ISPSLIB
      THASP.ISPF.ISPMLIB

      THASP.JCLIB             - contains JCL specific to the Production
                                environment, eg extracting data from the
                                Archive Tape Master file.

      THASP.REXX.EXEC         - contains utilities specific to the Production
                                environment

      THASP.SASLIB            - contains user-written SAS programs for
                                inclusion as 'Processes' in the Data
                                Retrieval Sub-system.
                              - see options S and V on panel PSEL

      THASP.SASUTIL           - contains other SAS programs

      THASP.UPGRADE.NOTICE    - when promotions are done, a notices should be
                                added to this PDS.
```

3.2 Compiling and Linking

JCL is NOT included with each PL/I module.

REXX programs have been created (in THASD.REXX.EXEC) to perform the various compile and link functions, in development and production libraries, in TSO foreground (demand) or in Batch.

These REXX utilities are started from the native TSO READY prompt, from PDS option 6, or from any PDF command prompt preceded by 'TSO '.

%PLI <environment> <module> /B

```
  where <environment> is: W - <userid> libraries
                          D - THASD libraries
                          P - THASP libraries
        <module>      is: - the module in SRCELIB you want compiled
                          - if shorter than 4, THAS will be prepended
        /B  (Brief) will suppress PL/I diagnostics output, so that
            only the PLI and LINK return codes will be displayed.

  e.g:  %PLI D 200    - will compile THAS200 in THASD libs.
                      - the compile listing will go in THASD.LISTLIB(THAS200)
```

%LINK <environment> <module>

```
  where <environment> is: W - <userid> libraries
                          D - THASD libraries
                          P - THASP libraries
        <module> is: - the module in OBJLIB you want linked
                     - if shorter than 4, THAS will be prepended

  e.g:  %LINK D 200    - will link THAS200 in THASD libs.
```

%PLIL <environment <module>

- does both %PLI and %LINK.

A member (module) name must be given after each of these commands. If the provided member name is less than 4 characters long, 'THAS' is automatically added as a prefix. For example, to compile and link program THAS030, enter the following TSO command:

```
      %PLIL D THAS030
 or   %PLIL D 030
```

The compile listings are written to PDS THASD.LISTLIB. The REXX program displays on the screen only the Diagnostics section of the compile listing. This is usually all that is needed to determine the cause of compile problems.

%DOLIST listfile cmda $ cmdb

```
    where: listfile       - name of file containing a list of strings
           cmda and cmdb  - any command text, may include blanks,
           $              - replaced by each string in the list_file.
```

DOLIST constructs and executes one TSO command for each record in listfile.

%BAT tsocmd

- submits tsocmd as a batch job.

The BAT and DOLIST utilities may be used to submit single or multiple compiles or links (or other commands) in batch:

Example:

```
  %BAT %DOLIST modlist %PLI D $ /B
```

- submits a batch job to compile all modules listed in file MODLIST

3.3 RACF permissions

RACF groups control what datasets and libraries you can look at and write to. It is only possible to belong to one RACF group at a time. The following RACF groups are relevant for H.A.S. development:

```
   THASD  - should be a developer's default group
          - alter (write) access to THASD files.

   THASPZ - use when doing promotions
          - read access to THASD and THWYD files, alter access to THASP
            files.

   THASP1 - H.A.S. user's default
   THASP  - more privileged (HQ) HAS users.
          - alter access to all THASP files except for the following files,
            to which read access only is allowed:
```

```
                THASP.SRCELIB
                THASP.OBJLIB
                THASP.LOADLIB
                THASP.PROCLIB
                THASP.TEXTLIB
                THASP.CARDLIB
                THASP.ISPF.*

        - no access to THASD files
        - developers should use this to test that systems work with the
          user's RACF.
```

To change your RACF group, use the following on the command line of any PDF menu:

```
          TSO CHGROUP group [password]
```

If the password is omitted, you will be prompted for it.

3.4 SYSEXEC Setup

In the PROCEDURE field of the TSO logon panel, insert SPFAUTO.
This causes CLIST(INIT) to be executed, but does NOT automatically start PDF.

Then insert something like the following in your CLIST(INIT) member.
(Only the bold statements are essential for HAS development.)

```
/* REXX */
/*====================================================================
|  INIT:       REXX Exec Invoked by SPFAUTO Logon Procedure.
|
|  Notes on SYSEXEC, SYSPROC, REXX execs and CLISTS:
|
|  Put REXX execs in one of the REXX.EXEC libraries allocated
|  to SYSEXEC.  At a TSO prompt they can be executed with %member.
|  After the SYSEXEC libraries are searched, the SYSPROC libraries
|  are searched.
|
|  Put CLISTS in userid.CLIST.  Start them with: EXEC (member)
|  (I tried allocating SC81171.CLIST to SYSPROC, but executing
|   %PDF removes it.  Conclusion: leave SYSPROC to the system,
|   use SYSEXEC for my own stuff.)
|
|  Note: TSO assumes that anything in a library named A.B.EXEC
|        is a REXX program, (i.e. the /* REXX */ comment is not
|        required) so putting a CLIST in it does not work.
|-------------------------------------------------------------------*/
"PROFILE WTPMSG"                  /* get more messages */
"EXECUTIL SEARCHDD(YES)"          /* TELLS SYSTEM TO LOOK IN SYSEXEC */
                                  /* FOR REXX EXECS NAMED "%NAME"    */
DSN = "REXX.EXEC 'THASD.REXX.EXEC' 'THASP.REXX.EXEC'"
"ALLOCATE DDNAME(SYSEXEC) SHR REUSE DSNAME("DSN")"
"ALLOCS SYSEXEC"
"EXEC 'THASD.REXX.EXEC(WHOSON)' EXEC"
```

3.5 REXX Utilities

PDS THASD.REXX.EXEC contains the following utilities which can be entered as TSO commands. Precede the names with a percent sign to avoid confusion with like-named TSO commands:

```
      BAT     -   submits a TSO command as a batch job.

      COPY    -   copies datasets to datasets, members to members, members to
                  datasets, datasets to members.  Automatically creates
                  target datasets if they don't exist.
              -   may not work with record types other than FB.
              -   see the internal documentation for details
              -     %COPY 'SYSCON.TEXT.TIB(D1994123)'  TIBTEMP

      DIR     -   lists members of a PDS, to the screen or to a file:
                    %DIR 'THASD.REXX.EXEC'          - lists members to screen
                    %DIR 'THASD.REXX.EXEC' MEMLIST  - lists to file MEMLIST

      DOLIST  -   repeats a TSO command for each record in a list.

      TYPE    -   types a file to the screen.  E.g.:
                    %TYPE 'THASD.JCLIB(THASJU20)'

      HEAD    -   types the first 10 lines of a file.  The number of lines
                  typed can be specified.  Issuing the command as
                    %HEAD file n
                  will type the first n lines of file 'file'.

      LINK    -   links a compiled PL/I program.

      PLI     -   compiles a PL/I module.

      PLIL    -   compiles and links a PL/I program

      PROMOTE -   copies members from THASD to THASP libraries:
              -   see next section for details.

      SUBUTIL -   allows you to type: SUBUTIL RUN944
                             instead of: SUBMIT 'THASD.UTILJCL(RUN944)'

      TAIL    -   types the last 10 lines of a file.  The number of lines
                  typed can be specified, as with %HEAD.

      TSOBR   -   browse a file from native TSO
                  (ISPF command B does the same thing from within PDF)

      TSOED   -   edit a file from native TSO
                  (ISPF command ED does the same thing from within PDF)

      WHOSON  -   determines which (if any) of a list of known users of
                  H.A.S. are currently logged on.
```

The following EXEC is an ISPF/EDIT macro:

```
      WYLCHA  -   simulates certain aspects of the Wylbur CHANGE command, for
                  use within other REXX EXECs (converted from Wylbur edit
                  scripts).
```

THASD.REXX.EXEC also contains the HAS ISPF Dialog startup programs.

3.6 Promoting Modules from Development to Production

RACF group THASPZ must be used for copying modules from THASD to THASP libraries.

Module promotions can be done in PDF, using option 3.3, but the PROMOTE utility in THASD.REXX.EXEC has been written to greatly simplify the process.

PROMOTE has the following features:

- allows member names and library names to be abbreviated,
- allows multiple module promotions in one command,
- checks the current RACF group, and if it is not THASPZ, a CHGROUP command is automatically issued (you will be prompted for your password),
- logs all promotions to file THASP.PROMOTE.LOG

The syntax of the command is as follows:

```
  %PROMOTE memlist liblist

      where memlist:  - one or more member names, separated with commas (no
                        spaces)
                      - member names may be abbreviated by omitting the
                        leading 'THAS'; e.g. 200 = THAS200
            liblist:  - one or more library IDs, separated by commas (no
                        spaces)
                      - library IDs are defined below


            Library             Library ID's        Example ID's
            SRCELIB             SRC*, PLI           SRC, SRCELIB, PLI
            OBJLIB              OBJ*                OBJ, OBJLIB
            LOADLIB             LOA*                LOA, LOAD, LOADLIB
            ISPF.ISPCLIB        ISPC*, REX*         REX, REXX, ISPC ISPCLIB
            ISPF.ISPPLIB        ISPP*, PAN          PAN, PANEL, ISPP,ISPPLIB
            ISPF.ISPSLIB        ISPS*, SK*          SK, SKL, SKEL, ISPS,
ISPSLIB
            PROCLIB             PR*                 PR, PROC, PROCLIB
            SASLIB              SASLIB,  SAS
            SASUTIL             SASUTIL

   Examples:
        1. promote panel THASP200:
            PROMOTE P200 PAN

        2. promote panel, rexx and skeleton THASP200:
            PROMOTE P200  PAN,REX,SKL

        3. promote PL/I modules THASAAA & THASBBB in SRCELIB & OBJLIB:
            PROMOTE AAA,BBB  SRC,OBJ

        4. promote PL/I main program THAS030 in SRCELIB, OBJLIB & LOADLIB:
            PROMOTE 030 SRC,OBJ,LOAD
```

3.7 Upgrade Notices

Whenever promotions to the production libraries are done, an 'Upgrade Notice' should be posted in PDS THASP.UPGRADE.NOTICE. Upgrade notices are numbered sequentially, with three digit numbers. The member name for an upgrade notice is the upgrade number preceded by an N.

After posting an upgrade notice, affected users and other developers should be notified by Email. They can then look at the notice if they are interested in the details.

Following is a sample upgrade notice:

```
                          HIGHWAY ACCIDENT SYSTEM

                               Upgrade Notice

Number      : 022
Notice Date : June 18, 1993

Upgrade Date: June 18, 1993
Upgrade Time: 13:06

Done By     : Matthew Nicoll

Description:

Fixed a bug which caused the HAS ISPF system to crash when attempting
to select data codes of the OCCUPY field.

Notes:

Changed the call sequence for the THASSMVB REXX program so that instead
of being supplied with the table name, it creates it internally, using
the TABID.

Modules Promoted:

THASP.ISPF.ISPCLIB: THASSMVB, THASP210, THASFSEL
```

3.8 PL/I Utility Programs

The following programs were written for development and testing utility purposes. They are in THASD.SRCELIB.

- JCL to execute most of these programs is in the library THASD.UTILJCL.
- some programs may be run at the TSO prompt using REXX execs in library THASD.REXX.EXEC
- PL/I source member names are THAS9xx.
- corresponding JCL or REXX members for execution are named RUN9xx.

900 - lists part of an accident file, given rec # range and case numbers
901 - print line where a field changes.
902 - read and check a SEGCLASS file.
903 - read and check a SEGDIST file.
909 - test THASJUL and THASYMD
910 - converts 256 byte records to 4 80 byte records, for custom editing.
911 - reverse of 910.
912 - creates 1 80 byte record, of key fields, from each 256 byte record.
913 - extracts the first 33 bytes from each record of two acc. files.
914 - compares most fields of 2 accident data files (for testing THAS010).
915 - compares location codes and obsolete_locn fields of corresponding records.
917 - test routine for subroutine THASZKM.
918 - extracts selected accident records from a VALLOC file.
919 - extracts MVB records with case #'s on coordinated HAS accident file
920 - extracts selected records from an MVB file.
921 - extracts selected fields from an MVB file
922 - creates test MVB records.
923 - prints fields of fatal accidents from an MVB file.
924 - test routine for subroutine THASTLU.
925 - test Critical value lookup (THASGCR, LCR, PCR) (not up-to-date).
926 - test THASGKD & THASLMK
927 - test THASLMD
928 - test THASGNV
929 - creates an HSBFATAL file from an accident file.
930 - creates a test accident file.
931 - a variant of 930.
932 - create a test accident file.
935 - copy selected records of an accident file.
936 - copies records of an accident file, where first and last record numbers to copy are read from SYSIN (e.g., 20, 200).
937 - copy selected years and segments of an accident file
938 - Split an accident file into two, based upon a list of segments read from SYSIN.
940 - checks date sequence of an accident file.
943 - counts total numbers of fatals, fatalities, injuries etc.
944 - Copies an accident file, producing an accident file with no more than one accident at each location.
945 - reads an accident file and counts accidents in each month and year.
946 - counts # fatal, injury & PDO accidents per location.
947 - counts attended accidents by year
948 - copies an accident file, copying non-node seg-kms to locn_id(2:9)
949 - modified THAS152 - does transformation on Burnaby segment 0720
950 - copies an accident file, eliminating records duplicated in another (smaller) accident file.
952 - copies an accident file, making specified changes to the data.

953 - copy without duplicates.
954 - tests THASVLN.
955 - coordinates an Accident file with another file, inserting the ATTENDED field.
956 - coordinates 2 acc files by case # and page, combines data for output.
957 - combines data of two one-to-one accident files.
958 - tests subroutine THASVOL
959 - test routine THASVSS.
960 - checks PAGE_NO, ACCASE & ACCASCON fields of an accident file.
962 - checks whether an accident file is in proper sequence (segment order).
964 - finds all locations with STOP or YIELD Traffic Control field.
965 - remove records from 964 output which have LMK type #1.
966 - reads and compares two sub-segment traffic volume files,
967 - checks TOTALKLD and TOTALINJ in multi-page accidents.
968 - checks month and hour fields
972 - sets a condition code specified as a parameter
976 - counts numbers of records and accidents and ensures that page-1 data matches subsequent pages of multi-page accidents.
977 - 1989 data conversion program.
978 - creates records for testing 977.
979 - compares files before and after running 977.
981 - restores to disk a set of update files that were archived on tape by Job U80. (JCL only, no source code needed.)
982 - extract DATE, CASE, LOCN_CODE, MVB_LOCN
983 - copy a file removing duplicates.
984 - copy a Surrey location code conversion table, transforming the LKI location codes from pre-95 to post-95 locations
985 - inserts region, district and locn_id.
986 - set LMK_NUM field in a landkmark file.
987 - reduce a LMK file to 1 record per rounded location.
988 - create a LKI VARIABLE record for each accident record.
989 - extract intersection accidents from an accident file.
990 - propagates Region, District and Locn_id throughout all accident records of each Group_id
991 - extracts records from a Landmark File.
992 - reads the Landmark file, and for each landmark, writes out a record with lmk type, highway class and traffic volume range.
993 - test routines GSC & LSC
994 - same as 992, using old routines ESC & SCT
995 - reads a PERM/SHORT.STATS.MASTER.STNYEAR file, and lists the 1st 80 bytes of records of specified years.
996 - check the sequence of a Landmark file.
997 - analyse output of programs 964 & 965
998 - make changes to a landmark file.
999 - test PL/I program for testing compiles.

3.9 Sorts and Merges

Two catalogued procedures, THASORTB and THASORTS, were created in THASP.PROCLIB, to do Big and Small sorts, respectively. They take a symbolic parameter defining the THASP.CARDLIB member which contains the control cards to be used. The first 5 characters of the CARDLIB member names for sorts are THASS. The symbolic parameter provides the last 3 characters.

A third catalogued procedure, THASMERG, does a merge. The first 5 characters of merge control members in CARDLIB are THASM.

The three character sort/merge codes are as described in the table below. The control card listings follow.

Sort/ Merge	Code		Primary Key	Secondary Keys
S	ACN	-	Accident Case	- PAGE
S/M	DAT	-	Date	- TIME, ACCIDENT CASE, PAGE
S	F4C	-	first 4 characters	
S	F7C	-	first 7 characters	
S	F8C	-	first 8 characters	
S	JPL	-	Jur, Police Code	- JUR, POLICE CODE, SEGMENT, KMMARK, DATE, TIME, CASE, PAGE
S/M	LCN	-	Location Code	- DATE, TIME, ACCIDENT CASE, PAGE
S	LKI	-	Lmk file by:	- SEGMENT, KMMARK, LMK_NUMBER, SIDE, TYPE
S	NOD	-	Node	- SEGMENT, DATE, TIME, ACCIDENT CASE, PAGE
S	PLC	-	Police Code	- SEGMENT, KMMARK, DATE, TIME, AC CASE, PAGE
S/M	SEG	-	Segment	- KMMARK, DATE, TIME, ACCIDENT CASE, PAGE
S	SG8	-	Segment	- same as SEG for a file with an 8-byte prefix
S	SEQ	-	HSNTAB sequence #	- start-KMMARK (SEGCLASS file only)
S	SQ2	-	HSNTAB sequence #	- start-KMMARK (SEGDIST file only)
S	320	-	cols 3-22	

```
DSN=THASP.CARDLIB(THASMDAT)
      ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
***** ****************************** TOP OF DATA ***************************
00001 * MERGE BY DATE, TIME, ACCIDENT CASE, PAGE
00002   MERGE FIELDS=(16,20,CH,A,1,1,CH,A)
00003   END
***** ****************************** END OF DATA ***************************


DSN=THASP.CARDLIB(THASMLCN)
      ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
***** ****************************** TOP OF DATA ***************************
00001 * MERGE BY LOCATION CODE, DATE, TIME, ACCIDENT CASE, PAGE
00002   MERGE FIELDS=(3,12,CH,A,16,20,CH,A,1,1,CH,A)
00003   END
***** ****************************** END OF DATA ***************************


DSN=THASP.CARDLIB(THASMSEG)
      ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
***** ****************************** TOP OF DATA ***************************
00001 * MERGE BY SEGMENT, KMMARK, DATE, TIME, ACCIDENT CASE, PAGE
00002   MERGE FIELDS=(7,8,CH,A,16,20,CH,A,1,1,CH,A)
00003   END
***** ****************************** END OF DATA ***************************


DSN=THASP.CARDLIB(THASSACN)
      ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
***** ****************************** TOP OF DATA ***************************
00001 * SORT BY ACCIDENT CASE, PAGE
00002   SORT FIELDS=(28,8,CH,A,1,1,CH,A)
00003   END
***** ****************************** END OF DATA ***************************


DSN=THASP.CARDLIB(THASSDAT)
      ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
***** ****************************** TOP OF DATA ***************************
00001 * SORT BY DATE, TIME, ACCIDENT CASE, PAGE
00002   SORT FIELDS=(16,20,CH,A,1,1,CH,A)
00003   END
***** ****************************** END OF DATA ***************************


DSN=THASP.CARDLIB(THASSF4C)
      ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
***** ****************************** TOP OF DATA ***************************
00001 * SORT BY THE FIRST 4 CHARACTERS (SEGMENT NUMBER IN SEGMENT INDEX)
00002   SORT FIELDS=(1,4,CH,A)
00003   END
***** ****************************** END OF DATA ***************************


DSN=THASP.CARDLIB(THASSF7C)
      ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
***** ****************************** TOP OF DATA ***************************
00001 * SORT BY THE FIRST 7 CHARACTERS (ACCASE IN THE HSBFATAL FILE)
00002   SORT FIELDS=(1,7,CH,A)
00003   END
***** ****************************** END OF DATA ***************************
```

```
 DSN=THASP.CARDLIB(THASSF8C)
       ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
 ***** ****************************** TOP OF DATA ***************************
 00001 * SORT BY THE FIRST 8 CHARACTERS (NODE PREFIX IN JOB THASJU85)
 00002   SORT FIELDS=(1,8,CH,A)
 00003   END
 ***** ****************************** END OF DATA ***************************


 DSN=THASP.CARDLIB(THASSLCN)
       ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
 ***** ****************************** TOP OF DATA ***************************
 00001 * SORT BY LOCATION CODE, DATE, TIME, ACCIDENT CASE, PAGE
 00002   SORT FIELDS=(3,12,CH,A,16,20,CH,A,1,1,CH,A)
 00003   END
 ***** ****************************** END OF DATA ***************************


 DSN=THASP.CARDLIB(THASSNOD)
       ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
 ***** ****************************** TOP OF DATA ***************************
 00001 * SORT A NODE PREFIXED FILE BY NODE, SEGMENT, DATE, TIME, CASE, PAGE
 00002   SORT FIELDS=(1,8,CH,A,15,4,CH,A,24,20,CH,A,9,1,CH,A)
 00003   END
 ***** ****************************** END OF DATA ***************************


 DSN=THASP.CARDLIB(THASSPLC)
       ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
 ***** ****************************** TOP OF DATA ***************************
 00001 * SORT BY POLICE CODE, SEGMENT, KMMARK, DATE, TIME, ACCIDENT CASE,
PAGE
 00002   SORT FIELDS=(105,4,CH,A,7,8,CH,A,16,18,CH,A,1,1,CH,A)
 00003   END
 ***** ****************************** END OF DATA ***************************


 DSN=THASP.CARDLIB(THASSSEG)
       ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
 ***** ****************************** TOP OF DATA ***************************
 00001 * SORT BY SEGMENT, KMMARK, DATE, TIME, ACCIDENT CASE, PAGE
 00002   SORT FIELDS=(7,8,CH,A,16,20,CH,A,1,1,CH,A)
 00003   END
 ***** ****************************** END OF DATA ***************************


 DSN=THASP.CARDLIB(THASSSEQ)
       ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
 ***** ****************************** TOP OF DATA ***************************
 00001 * SORT BY HSNTAB SEQUENCE NUMBER AND START-KMMARK (SEGCLASS FILE)
 00002   SORT FIELDS=(48,4,CH,A,6,8,CH,A,24,7,CH,A)
 00003   END
 ***** ****************************** END OF DATA ***************************


 DSN=THASP.CARDLIB(THASSSG8)
       ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
 ***** ****************************** TOP OF DATA ***************************
 00001 * SORT A PREFIXED FILE BY SEGMENT, KMMARK, DATE, TIME, CASE, PAGE
 00002   SORT FIELDS=(15,8,CH,A,24,20,CH,A,9,1,CH,A)
 00003   END
 ***** ****************************** END OF DATA ***************************
```

```
DSN=THASP.CARDLIB(THASSSQ2)
       ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
 ***** ***************************** TOP OF DATA ************************
 00001 * SORT BY HSNTAB SEQUENCE NUMBER AND START-KMMARK (SEGDIST FILE)
 00002   SORT FIELDS=(31,3,CH,A,9,7,CH,A)
 00003   END
```

4 Master Files

4.1 Master Accident Files

The Highway Accident System maintains master files of three types of accident records:

PROV.VALLOC	jurisdiction 1 records with valid location codes
PROV.INVLOC	jurisdiction 1 nonfatal records without valid location codes
PROV.INVLOC.FATAL	jurisdiction 1 fatal records without valid location codes
MUN	jurisdiction 2 and 3 (municipal & rural) records

For each of these four file types, there are two files:

CURRENT - containing the current data.
ARCHIVE - containing the data prior to the CURRENT file.

The break date between the Current and Archive files is the same for each of the three file types.

Valid Master file names may be constructed as follows:

```
        | PROV.VALLOC.       | CURRENT | (0)
THASP.  | PROV.INVLOC.       | ARCHIVE | (-1)
        | PROV.INVLOC.FATAL.|          | (-2)
        | MUN.               |          |
```

e.g.:
THASP.PROV.VALLOC.CURRENT(0) - most up-to-date file of good provincial jurisdiction records.
THASP.MUN.CURRENT(-1) - municipal records prior to the last update.

Sort order:

PROV.VALLOC files: Location Code, Date, Time, Accident Case Number, Page.
Others: Date, Time, Accident Case Number, Page

These files are defined with the TMM Storage Management Class, which means that they are kept in compressed form on cartridge tapes. They are available in ISPF/PDF: if, for example, you wanted to Browse one of them, you would have to wait a minute or two (or five!) while it is recalled to disk storage. Within an hour of being used, the file will be automatically returned to cartridge tape.

Each file is maintained as a Generation Data Group, with three generations kept. This means that for each file, the latest and two previous versions of the files are kept.

Note that the PROV.INVLOC.FATAL file contains no data prior to 1999, since invalid location codes on fatal jurisdiction 1 accidents were not tolerated until then.

The location codes in the PROV.VALLOC files are according to the LKI in effect at the time of the accident.

4.2 Backups

Backup copies of the master accident files are created after each update (by Update job U40). They are named as follows:

```
CURRENT: THASP.UAyyyymm.PROV.VALLOC.CURRENT
         THASP.UAyyyymm.PROV.INVLOC.CURRENT
         THASP.UAyyyymm.PROV.INVLOC.FATAL.CURRENT
         THASP.UAyyyymm.MUN.CURRENT
```

- where yyyymm is the Update date, i.e. the month of the latest accident date on the file.
- these files are created with the TMM3Y Storage Management Class, which means they are automatically deleted by the system three years after they are created.

The following backup files are created (by Update job U45) after old data is transferred from the CURRENT to the ARCHIVE files:

```
CURRENT: THASP.UAJOBU45.PROV.VALLOC.CURRENT
         THASP.UAJOBU45.PROV.INVLOC.CURRENT
         THASP.UAJOBU45.PROV.INVLOC.FATAL.CURRENT
         THASP.UAJOBU45.MUN.CURRENT
```

- these are permanent TMM files.

```
ARCHIVE: THASP.UAyyyy.PROV.VALLOC.ARCHIVE
         THASP.UAyyyy.PROV.INVLOC.ARCHIVE
         THASP.UAyyyy.PROV.INVLOC.FATAL.ARCHIVE
         THASP.UAyyyy.MUN.ARCHIVE
```

- where yyyy is the year of last year of data on the files.
- these are TMM3Y files - deleted by the system after three years.

4.3 File Sizes

File Sizes, October 2001

	CURRENT		ARCHIVE	
	Records	M-bytes	Records	M-bytes
PROV.VALLOC	269741	66	180000	46
PROV.INVLOC	80748	20	125000	31
PROV.INVLOC.FATAL	12			
MUN	877267	215	741000	190
Years of Data:	1987-2000/06		78-86	

4.4 Accessing Master Files

The Tape Master files:

- have accidents added by Update job U40.
- can have old accidents moved from CURRENT to ARCHIVE by Update job U45
- are read for Production (PDS) Master File creation by Utility job PDSM.
- can be corrected using an option on the System Utilities menu.

4.5 PDS Master

The "PDS Master" consists of a pair of Partitioned Data Sets. They contain the data from the THASP.PROV.VALLOC.CURRENT(0) tape master file, with some additions and modifications. The PDS's form the accident database organized for efficient access by H.A.S. Data Retrieval sub-system.

The two PDS's are:

`THASP.SEG.ACDAT`

- contains all accidents which are NOT at Nodes
- each member contains the data of one segment. All the data of segment 1234 is stored in member SEG1234
- records are sorted by KMMARK, Date, Time and Case

`THASP.NODE.ACDAT`

- contains all accidents which are at Nodes
- each member contains the data of one Node.
- a member name is the node name with the first character translated from a digit to a letter as follows: 0,1,2,...,9 => Z,A,B,...,I
- records are sorted by Segment, Date, Time and Case

Utility job PDSM is used to create the PDS Master. In addition to organizing the data, job PDSM makes the following modifications:

- transforms location codes to reflect changes in the LKI definitions which have occurred since the accident was recorded.
- clears the LOCN_ERROR flags,
- replaces the MVB_LOCN field with the LOCN_ID,
- looks up the Region and District for each accident, and inserts them into the accident records,
- calculates and inserts the Day-of-Week.

Job PDSM also creates a file called THASP.PDSM.DATELIMS, which contains the first and last accident dates of the PDS-Master. (This file becomes THASP.PDSMAST.DATELIMS when job PDSI is run.)

4.6 Archive PDS Master

The Archive PDS-Master files are the equivalent of the PDS-Master, created from the THASP.PROV.VALLOC.ARCHIVE(0) tape master file.

The Archive PDS-Master files are named:

`THASP.ARC.SEG.ACDAT`
`THASP.ARC.NODE.ACDAT`
`THASP.ARC.PDSMAST.DATELIMS`

These files can not be created from the H.A.S. menu system. They only need to be created when the starting year of the PDS-Master is changed. They are created by submitting THASP.JCLIB(ARCPDSM).

5 File and Record Descriptions

5.1 Accident Data Files

5.1.1 Accident Data File Description

The Current and Archive Master files, and most of the Update files are 'Accident Data' files. They all have 320 byte records (defined in the following section), and are fixed block sequential.

There is one record for each MV6020 (originally MV104) form. One MV6020 is filled out for each two vehicles of one accident. If there is more than one page filled out for one accident, each page becomes one record. The pages are numbered 1, 2, ... up to 9. In some of the Archive Master Files, due to problems of bad data, the old HSI system, and sorting, there are many records with page numbers greater than 1 which are not immediately following their correct page 1 record! Checks built into the H.A.S. system now prevent this from happening. Sorts of the files must always take the multi-page accidents into consideration.

The PROV.VALLOC (Provincial accidents with valid Location Codes) Current and Archive files are sorted in Location Code order (primarily). The other Current and Archive Master files are sorted by Accident Date. The Accident Data files of an Update are sorted in a variety of orders - the easiest way to determine the sort order of a particular Update file is to look at the system flowcharts.

5.1.2 Numeric and Letter Code Fields

Some fields are stored differently in the H.A.S. accident record than how they were coded on the original MVB form. Some of the changes were made to save space; others separate the individual digits of a two digit codes.

All fields on the MVB form may be coded as 'data not present'. There are three different 2-digit codes for this:

```
00          UNKNOWN
98          NOT APPLICABLE
99          OTHER
```

There are no 1-digit 'data not present' codes on the MVB form; however, some of the 1-character fields in the H.A.S. accident record require 'data not present' codes of X, Y, and Z (corresponding to 98, 99, and 00) as illustrated in the three sections below.

5.1.2.1 SPEEDTYP and SPEEDLIM

The fields SPEEDTYP and SPEEDLIM come from the 3-digit MVB Speed Zone field. The first digit of the MVB field gives the type of speed zone, and the 2nd and 3rd digits together give the speed limit. The decoding is done as follows:

Speed Zone	SPEEDTYP	SPEEDLIM	
'98 '	X	X	
'99 '	Y	Y	
'00 '	Z	Z	
'ab '	a	b	(where a is 1-3, b is 1-9)
'a10'	a	A	
'a11'	a	B	
'a99'	a	Y	

If there are ever speed limits beyond 110 km/hr, Speed Zone will be 'a12', 'a13', and so on, and SPEEDLIM can be coded C or D for 120 or 130 km/hr., etc.

5.1.2.2 Two-to-one Character Fields

Certain fields were coded as 2-digit fields on the MVB form, but are stored in only one character in the Accident Record. These fields are:

ROADSURF
DAMSVRT1
DAMSVRT2
ROADTYPE
LANDUSE
WEATHER
LIGHTING
LOCN1ST
PEDLOCN
TRAFFLOW

And in the Victim Table: SAFQUIP, EJECTIO, VICTCON

These fields are coded in the H.A.S. Accident Record as follows:

MVB field	HAS field	
'98'	X	
'99'	Y	
'00'	Z	
'0a'	a	(where a is 1-9)

5.1.2.3 Split Fields

The following 2-digit MVB fields have been split into pairs of 1-character fields in the H.A.S. Accident Record:

```
Road Class          ->  NUMLANE   ROADCLAS  (stored in reverse order)
Roadway Character   ->  ROADCURV  ROADGRAD  (stored in same order)
```

These fields are coded in the Accident Record as follows:

MVB field	HAS field 1	HAS field 2	
'98'	X	X	
'99'	Y	Y	
'00'	Z	Z	
'ab'	a	b	(where a is 1-9, b is 1-9)

5.1.3 Accident Data Record Description

Included in this section is a detailed description of each field, followed by the PL/I record definition.

Most fields come from the accident report (form MV6020) via the MVB file. MV6020 template field numbers are provided where appropriate. For details on how the MV6020 is completed, see the TRAFFIC INCIDENT REPORTING POLICE PROCEDURES MANUAL.

"PDS-Master only" means that the field is inserted by program THAS110 in utility job PDSM, and is thus present in the PDS-Master files but not in the tape master files.

"Post-THAS200" means that the field is inserted by program THAS200, and is thus only present in post-data-selection accident subsets.

Latest revision: April 2004

BYTE	NAME	LENGTH	DESCRIPTION
1	PAGE_NO	01	- MV6020 page number - more than 1 page is required if there are more than 2 vehicles involved in the accident
2	JURCODE	01	- jurisdiction code - numeric: 1-Provincial, 2-Municipal, 3-Rural
3	LOCN_CODE	12	- the Location Code consists of the following 4 fields:
3	HIGHNUM	03	- highway number (does not include the letter) - numeric, right justified, blank filled
6	HIGHLET	01	- highway letter (There are 3 spaces on the MV6020, and in the MVB file, for the highway number and optional letter.

			Program THAS010 extracts the letter into HIGHLET, if there is one.)
7	SEGNUM	04	- highway segment number of the accident location - numeric, zero filled - second section of 'Location Code' on the MV6020 - segment numbers are defined in the LANDMARK KILOMETRE INVENTORY
11	KMMARK	04	- kilometre mark of the accident location - numeric - km to 1 decimal place (no decimal point) - last section of the 'Location Code' on the MV6020
15	OBSOLETE_LOCN	01	- X if the accident occurred on an obsolete section of highway; blank otherwise - used to be FILLKM (the old HSI system had a 2nd decimal place for the KMMARK)
16	ACCDATE	08	- accident date, yyyymmdd - numeric: year, month day
24	ACCHOUR	04	- accident time - numeric - 24 hour time
28	ACCASE	08	- case number - sequential on multiple page accidents prior to 1984 - now same case number on all pages of an accident from 1984 - alphanumeric
36	POLICEFILE	16	- police file number. Length varies. Most are 8.
52	TOTALINJ	03	- total number of people injured in the accident - numeric
55	TOTALKLD	03	- total number of people killed in the accident - numeric
58	TOTALVEH	03	- total number of vehicles involved in the accident - numeric
61	SPEEDTYP	01	- speed zone type (Posted, or Special) - numeric (1 or 3) - 1st digit of #2A on the MV6020

62 SPEEDLIM 01 - speed limit code
- 1-9, A,B,C
- from 2nd and 3rd digits of #2A on the MV6020

63 SPEEDADV 01 - advisory speed limit
- 1-9, A,B,C
- from 2nd and 3rd digits of #2B on the MV6020

64 LOCN_TYPE 02 - accident location-type code
- numeric
- #2 on the MV6020

66 TRAFCNTL 02 - traffic control code
- numeric
- #5 on MV6020

68 TRAFFLOW 01 - traffic flow code (one way or two way)
- numeric or letter code
- #1A on the MV6020

69 ROADCLAS 01 - road class
- numeric (1, 2, or 3) or letter code (X, Y, or Z)
- 2nd digit of #1 on the MV6020

70 NUMLANE 01 - number of lanes
- numeric or letter code
- 1st digit of #1 on the MV6020

71 ROADSURF 01 - road surface condition code (dry, wet, muddy etc.)
- numeric or letter code
- #7 on the MV6020

72 ROADTYPE 01 - road type code (eg asphalt, gravel etc.)
- numeric or letter code
- #4 on the MV6020

73 ROADCURV 01 - road curve code (misspelled ROADCURC on report R1-2)
- numeric or letter code
- 1st digit of #6 on the MV6020 (Roadway Character)

74 ROADGRAD 01 - road gradiant code
- numeric or letter code
- 2nd digit of #6 on the MV6020 (Roadway Character)

75	LANDUSE	01	- land usage in accident area - numeric or letter code - #3 on MV6020
76	WEATHER	01	- weather condition code - numeric or letter code - #8 on the MV6020
77	LIGHTING	01	- lighting code - numeric or letter code - #9 on the MV6020
78	DIAGRAM	02	- number of the diagram describing the accident - numeric
80	LOCN1ST	01	- location of first contact - numeric or letter code - #24 on the MV6020
81	ATTENDED	01	- 1 if police attended the accident, 2 otherwise
82	POLICECD	04	- police code - numeric - see appendix A of the TRAFFIC INCIDENT REPORTING POLICE PROCEDURES MANUAL
86	PEDLOCN	01	- pedestrian location code - numeric or letter code - #29 on the MV6020

```
87   PEDNACTN      02  -  pedestrian action code
                       -  numeric
                       -  #30 on the MV6020

89   VEHTYPE1      02  -  type of vehicle 1
                       -  numeric
                       -  #27 on the MV6020

91   VEHTYPE2      02  -  type of vehicle 2
                       -  numeric
                       -  #28 on the MV6020

93   VEHUSAG1      02  -  usage of vehicle 1
                       -  numeric
                       -  #27A on the MV6020

95   VEHUSAG2      02  -  usage of vehicle 2
                       -  numeric
                       -  #28A on the MV6020

97   STOLEN1       01  -  Y or N
98   STOLEN2       01  -  Y or N

99   LIC_CLASS1    03  -  Licence Class of driver 1
102  LIC_CLASS1    03  -  Licence Class of driver 2

105  NOVLEARN1     01  -  N, L or blank (Novice, Learner) driver 1
106  NOVLEARN2     01  -  N, L or blank (Novice, Learner) driver 2

107  PREACTN1      02  -  pre-collision action of vehicle 1
                       -  numeric code
                       -  ##25 on MV6020

109  PREACTN2      02  -  pre-collision action of vehicle 2
                       -  numeric
                       -  #26 on the MV6020

111  TYPE2ND1      02  -  second event code for vehicle 1
                       -  numeric
                       -  #21 of the MV6020

113  TYPE3RD1      02  -  third event code for vehicle 1
                       -  numeric
                       -  #22 on the MV6020
```

```
115  TYPE3RD2      02  -  third event code for vehicle 2
                       -  numeric
                       -  #23 on the MV6020

117  CONTRB11      02     up to 4 contributing factors to the accident for
119  CONTRB12      02  /  vehicle 1.
121  CONTRB13      02  \  numeric.
123  CONTRB14      02     #31, 32, 33 & 33A on the MV6020


125  CONTRB11      02     up to 4 contributing factors to the accident for
127  CONTRB12      02  /  vehicle 2.
129  CONTRB13      02  \  numeric.
131  CONTRB14      02     #34, 35, 36 & 36A on the MV6020

1334 VEHDIR1       01  -  the direction of travel for vehicle 1
                       -  N, E, W, S, P (parked) or U (unknown)
                       -  unnumbered MV6020 field
134  VEHDIR2       01  -  the direction of travel for vehicle 2

135  DAMSVRT1      01  -  damage severity code of the first vehicle
                       -  numeric or letter code
                       -  codes (1 to 5) explained on back of the MV6020
                          template

136  DAMSVRT2      01  -  damage severity code of the second vehicle

137  GROUP_ID      04  -  used to divide accidents in a subset into groups
                          (e.g., a group may be all accidents in one
                          accident-prone section)
                       -  set and used in the Data Retrieval sub-System

141  HWYCLASS      08  -  Highway Classification
                       -  set in THAS200

149  WEEKDAY       01  -  day of the week
                       -  numeric
                       -  Monday is 1, Sunday is 7
                       -  set in PDS-Master, by program THAS110
150  MVB_LOCN      11  -  the location code as copied from the MVB file
                          (unmodified) -- only in tape file (pre-PDS Master)

150  LOCN_ID       09  -  overlaid on MVB_LOCN in PDS Master files
                       -  defined in THASRACO
```

- the Location ID consists of the following 2 fields:

150 LOCN_ID_TYPE 01
- letter code (N or S)
- indicates whether accident occurred at a node (N) or within a segment (S)

151 LOCN_ID_CODE 08
- nodename if accident occurred at a node
- segment + kmmark otherwise (same as SEGNUM and KMMARK fields above)
- numeric

161 LOCN_ERROR 01
- 8 bits, set by program THAS030, indicating the following:
  1 - highway number/letter is bad
  2 - segment number is invalid
  3 - highway - segment mismatch
  4 - KMARK invalid for this segment
  5 - location type does not match LKI
  6 - police code does not match LKI
  7 - LOCN modified to make it valid
  8 - LOCN modified but still not valid
- used in the Update sub-system only.
- set blank by program THAS101 when PDSMASTER is created.

162 REGION 01
- highway region
- numeric
- PDS-Master only

163 DISTRICT 02
- highway district
- numeric
- PDS-Master only

165 AREA 02
- Contract Management Area
- numeric
- PDS-Master only
* NOT IMPLEMENTED AS OF 2004-04-14 *

167 VEH_WT 02
- vehicle causal factor
- a percent, range 00 to 99
- calculated by THAS140 in job U30

169 ROAD_WT 02
- road causal factor
- a percent, range 00 to 99

```
                        -  calculated by THAS140 in job U30

171  SEVERITY_TYPE 01   -  1 - Fatal, 2 - Injury, 3 - PDO
                        -  PDS-Master only.
                        -  calculated from TOTALKLD and TOTALINJ
                           by program THAS110 in job PDSM.
```

VICTIM TABLE (# 10-20 on MV6020) - repeats 8 times

185 OCCUPY 02 - location of person, eg vehicle number, bicycle etc.
- numeric code
- #10 on the MV6020

187 POSNVEH 02 - position of person within or on a vehicle
- numeric code
- #11 on the MV6020

189 SAFQUIP 01 - safety equipment used
- numeric code
- #12 on the MV6020

191 EJECTIO 01 - code indicating whether or not the person was ejected from the vehicle.
- numeric code
- #13 on the MV6020

192 VICTAGE 02 - age of the person (victim)
- numeric
- #14 on the MV6020

194 VICTSEX 01 - sex of the person
- 1 letter, M or F
- #15 on the MV6020

195 LOCNINJ 02 - location of the person's most severe injury
- numeric code
- #16 on the MV6020

197 TYPEINJ 02 - type of injury
- numeric code
- #17 on the MV6020

199 INJCLASS 02 - injury class
- 01 - minor, 02 - serious
- #20A on the MV6020

201 VICTCON 01 - victim's state of consciousness
- numeric code
- #18 on the MV6020

```
  DATASET:   THASP.TEXTLIB
  MEMBER:    THASRACD

----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+

       2 ACDAT,
         5 PAGE_NO           CHAR(01),   /* PAGE NUMBER               */
         5 JURCODE           CHAR(01),   /* JURISDICTION CODE         */
         5 LOCN_CODE,                    /* LOCATION CODE             */
            10 HIGHNUM       CHAR(03),   /* HIGHWAY NUMBER            */
            10 HIGHLET       CHAR(01),   /* HIGHWAY LETTER            */
            10 SEGNUM        CHAR(04),   /* SEGMENT NUMBER            */
            10 KMMARK        PIC'999V9', /* KILOMETER MARK            */
         5 OBSOLETE_LOCN     CHAR(01),   /* X IF LOCN_CODE IS OBSOLETE*/
         5 ACCDATE           CHAR(08),   /* ACCIDENT DATE        2kdh */
         5 ACCHOUR           CHAR(04),   /* ACCIDENT TIME             */
         5 ACCASE            CHAR(08),   /* CASE NUMBER               */
         5 POLICEFILE        CHAR(16),   /* POLICE FILE NUMBER        *2004*/
         5 TOTALINJ          CHAR(03),   /* TOTAL INJURED             */
         5 TOTALKLD          CHAR(03),   /* TOTAL KILLED              */
         5 TOTALVEH          CHAR(03),   /* TOTAL VEHICLES            */
         5 SPEEDTYP          CHAR(01),   /* SPEED ZONE TYPE           */
         5 SPEEDLIM          CHAR(01),   /* SPEED LIMIT CODE          */
         5 SPEEDADV          CHAR(01),   /* Advisary speed limit      *2004*/
         5 LOCN_TYPE         CHAR(02),   /* ACCIDENT LOCATION         */
         5 TRAFCNTL          CHAR(02),   /* TRAFFIC CONTROL CODE      */
         5 TRAFFLOW          CHAR(01),   /* TRAFFIC FLOW CODE         */
         5 ROADCLAS          CHAR(01),   /* ROAD CLASS                */
         5 NUMLANE           CHAR(01),   /* NUMBER OF LANES           */
         5 ROADSURF          CHAR(01),   /* ROAD SURFACE CONDITION    */
         5 ROADTYPE          CHAR(01),   /* ROAD TYPE CODE            */
         5 ROADCURV          CHAR(01),   /* ROAD CURVE CODE           */
         5 ROADGRAD          CHAR(01),   /* ROAD GRADIANT CODE        */
         5 LANDUSE           CHAR(01),   /* LAND USAGE                */
         5 WEATHER           CHAR(01),   /* WEATHER CONDITION CODE    */
         5 LIGHTING          CHAR(01),   /* LIGHTING CONDITION CODE   */
         5 DIAGRAM           CHAR(02),   /* DIAGRAM CODE NUMBER       */
         5 LOCN1ST           CHAR(01),   /* LOCATION OF 1ST CONTACT   */
         5 ATTENDED          CHAR(01),   /* 1=ATTENDED, 2=NOT ATT.    */
         5 POLICECD          CHAR(04),   /* POLICE CODE               */
         5 PEDLOCN           CHAR(01),   /* PEDESTRIAN LOCATION CODE  */
         5 PEDNACTN          CHAR(02),   /* PEDESTRIAN ACTION CODE    */

                                         /**** Vehicle Information ****/
         5 VEHTYPE1          CHAR(02),   /* VEHICLE 1 TYPE            */
```

```
   5 VEHTYPE2          CHAR(02),   /* VEHICLE 2 TYPE            */
   5 VEHUSAG1          CHAR(02),   /* VEHICLE 1 USAGE           */
   5 VEHUSAG2          CHAR(02),   /* VEHICLE 2 USAGE           */
   5 STOLEN1           CHAR(01),   /* Y or N                    *2004*/
   5 STOLEN2           CHAR(01),   /* Y or N                    *2004*/
   5 LIC_CLASS1        CHAR(03),   /* Licence class of driver 1 *2004*/
   5 LIC_CLASS2        CHAR(03),   /* Licence class of driver 2 *2004*/
   5 NOVLEARN1         CHAR(01),   /* N, L or blank             *2004*/
   5 NOVLEARN2         CHAR(01),   /* N, L or blank             *2004*/
   5 PREACTN1          CHAR(02),   /* VEHICLE 1 PRE-COLLISION   */
   5 PREACTN2          CHAR(02),   /* VEHICLE 2 PRE-COLLISION   */
   5 TYPE2ND1          CHAR(02),   /* VEHICLE 1 SECOND EVENT    */
   5 TYPE3RD1          CHAR(02),   /* VEHICLE 1 THIRD EVENT     */
   5 TYPE3RD2          CHAR(02),   /* VEHICLE 2 THIRD EVENT     */
   5 CONTRB11          CHAR(02),   /* UP TO 4 CONTRIB  FACTORS  */
   5 CONTRB12          CHAR(02),   /*   TO THE ACCIDENT FOR     */
   5 CONTRB13          CHAR(02),   /*   VEHICLE 1               */
   5 CONTRB14          CHAR(02),   /*                           *2004*/
   5 CONTRB21          CHAR(02),   /* UP TO 4 CONTRIB  FACTORS  */
   5 CONTRB22          CHAR(02),   /*   TO THE ACCIDENT FOR     */
   5 CONTRB23          CHAR(02),   /*   VEHICLE 2               */
   5 CONTRB24          CHAR(02),   /*                           *2004*/
   5 VEHDIR1           CHAR(01),   /* VEHICLE 1 DIRECTION       */
   5 VEHDIR2           CHAR(01),   /* VEHICLE 2 DIRECTION       */
   5 DAMSVRT1          CHAR(01),   /* DAMAGE SEV , 1ST VEHICLE  */
   5 DAMSVRT2          CHAR(01),   /* DAMAGE SEV , 2ND VEHICLE  */

                                   /*** Calculated/Others *******/
   5 GROUP_ID          CHAR(04),   /* USED IN DATA RETR. SYSTM  */
   5 HWYCLASS          CHAR(08),   /* after selection only      *2004*/
   5 WEEKDAY           CHAR(01),   /* MONDAY=1 (in PDSM only)   */
   5 MVB_LOCN          CHAR(11),   /* LOCATION FROM MVB (TAPE)  */
                                   /* LOCN_ID OVERLAYED (PDSM)  */
   5 LOCN_ERROR        BIT(8)  ,   /* LOCATION ERROR FLAGS      */
   5 REGION            CHAR(01),   /* IN PDS-MASTER ONLY        */
   5 DISTRICT          CHAR(02),   /* IN PDS-MASTER ONLY        */
   5 AREA              CHAR(02),   /* IN PDS-MASTER ONLY        *2004*/
   5 VEH_WT            PIC'99',    /* Vehicle Causal Weight     */
   5 ROAD_WT           PIC'99',    /* Road Causal Weight        */
   5 SEVERITY_TYPE     CHAR(01),   /* 1:Fatal 2:Injury 3:PDO    *2004*/
   5 FILLER            CHAR(13),   /*                           *2004*/
                                   /**** Victim Information *****/
   5 VICTIM_TABLE(8),
     10 OCCUPY         CHAR(02),   /* LOCATION OF PERSON        */
     10 POSNVEH        CHAR(02),   /* POSITION OF PERSON        */
```

```
10 SAFQUIP      CHAR(02),   /* SAFETY EQUIPMENT USED     *2004*/
10 EJECTIO      CHAR(01),   /* EJECTION CODE             */
10 VICTAGE      CHAR(02),   /* VICTIM AGE                */
10 VICTSEX      CHAR(01),   /* VICTIM SEX                */
10 LOCNINJ      CHAR(02),   /* MOST SEVERE INJURY LOCN   */
10 TYPEINJ      CHAR(02),   /* TYPE OF INJURY            */
10 INJCLASS     CHAR(02),   /* INJURY CLASS              *2004*/
10 VICTCON      CHAR(01);   /* VICTIM CONDITION          */
```

```
  DATASET:   THASP.TEXTLIB
  MEMBER:    THASRACO

----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+

 /* ----------- Overlay Definitions of the Accident Record -------    */

    DCL (ACDAT_LOCN   CHAR(12),         /* Entire Location Code       */
         ACDAT_HNL    CHAR(4),          /* Highway Number and Letter  */
         SEG_KM       CHAR(8)  POS(5)   /* Segment and KMMARK         */
        )DEF ACDAT.LOCN_CODE;

    DCL  ACDAT_CHARKM CHAR(4)  DEF ACDAT.KMMARK;

    DCL (LOCN_ID      CHAR(9)  POS(1),  /* both of the following:     */
         LOCN_ID_TYPE CHAR(1)  POS(1),  /* S or N: Segment or Node    */
         LOCN_ID_CODE CHAR(8)  POS(2)   /* seg+km or node name        */
        )DEF ACDAT.MVB_LOCN;

    DCL  #KILLED      PIC'999'  DEF ACDAT.TOTALKLD;
    DCL  #INJURED     PIC'999'  DEF ACDAT.TOTALINJ;
    DCL  #VEHICLES    PIC'999'  DEF ACDAT.TOTALVEH;

                                /* During Update Only:                */
    DCL ($BADHWY      POS(1),   /* Highway number/letter is bad       */
         $BADSEG      POS(2),   /* Segment number is invalid          */
         $BAD_HWY_SEG POS(3),   /* Highway-segment mismatch           */
         $BADKM       POS(4),   /* Kmmark is invalid for this segment */
         $BAD_LOC_TYP POS(5),   /* Location type does not match LKI   */
         $BAD_POLICE  POS(6),   /* Police code does not match LKI     */
         $PARSE_VALID POS(7),   /* Locn modified to make it valid     */
         $PARSE_INVAL POS(8)    /* Locn modified but still not valid  */
        )BIT(1) DEF ACDAT.LOCN_ERROR;
```

*5.2 <userid>.ACCTYPE.RATIOS.**

Given a user-supplied *name*, program THAS270 can create a pair of Average Accident Type Ratio files, named:

```
userid.ACCTYPE.RATIOS.LOCATION.name
userid.ACCTYPE.RATIOS.SECTION.name
```

These files are used by programs THAS210 (Accident Prone Locations) and THAS220 (Accident Prone Sections) for the Counter-Measure method of identifying accident prone locations/sections. The user can create multiple such file pairs.

Note: in May 2001 these files replaced a single system-wide pair of files which were named THASP.LOCATION/SECTION.ACCTYPE.RATIOS.

Note: in April 2002, the Class Set field was lengthened from 8 to 20 characters, lengthening the record from 132 to 144. The PL/I routine which reads this file: THASATR, will read either the old 132-byte record files or the new 144-byte record files. Thus the old files do not need to be modified.

```
Sort Order: none.

 Columns  Description

   1-25     Name - description of the reference group defined by this record.

  27-46     Highway Classification set,

  47-63     Landmark Types: up to 6 2-character codes separated by single
            blanks
            - for the SECTION file, this field will always be blank.

  65-144    Average ratios for accident types 1 to 13, format 9.999

The following PL/I record definition is in THASP.TEXTLIB(THASRATR) :
(The old record definition is in THASP.TEXTLIB(THASROTR) )

                         /*--- AVERAGE ACCIDENT TYPE RATIOS Record ----*/

    DCL ATR_REC CHAR(144);
    DCL 1 ATR DEF ATR_REC,
        5 NAME      CHAR(25),           /* description of ref. group    */
        5 FILL1     CHAR(1),
        5 CLASS_SET CHAR(20),            /* highway classification set   */
        5 LMK_TYPE(6),                  /* up to 6 lmk type codes       */
          8 FILL    CHAR(1),
          8 CODE    CHAR(2),
        5 ACC_TYPE(13),                 /* 13 acc type ratios           */
          8 FILL    CHAR(1),
          8 RATIO   PIC'9V.999';        /* ratio                        */
```

*5.3 THASP.ARC.**

Files of the Archive PDS-Master and temporary intermediate files used in the creation of the Archive PDS-Master files, are named with the THASP.ARC high level indexes.

The Archive PDS-Master files are named:

```
THASP.ARC.SEG.ACDAT
THASP.ARC.NODE.SEG.ACDAT
THASP.ARC.PDSMAST.DATELIMS
```

These files are created with JCL THASP.JCLIB(ARCPDSM), and in development, THASD.UTILJCL(ARCPDSM). The intermediate files can be removed using JCL in member DARCPDSM in the same JCL libraries.

*5.4 THASP.AVERATES.**

These files store Average Accident Rates and Road-Weighted Average Accident Rates for Locations and Sections by Highway Classification Set and Landmark Types.

```
Sort Order: none.
```

There are separate files for Location and Section rates.
There may be multiple public and personal files, named as follows:

```
hli.AVERATES.LOC.name
hli.AVERATES.SEC.name
```

Where "hli" is THASP for public files, and a user's TSO userid for personal files. "name" is user-specified.
There is a pair of master public files, named as follows:

```
THASP.AVERATES.LOC.MASTER
THASP.AVERATES.SEC.MASTER
```

... which can be used as a common standard set of average rates.

These files are created by manually editing them on the mainframe. The data for the files can be obtained by running the Rate Table program. There are facilities on panel PARF - which is used for average rate file selection - for editing, copying and naming average rate files.

<u>File Formats</u>

There are three different file formats accepted (by PL/I program THASLAR) for these files:

1. Old format fixed length record, as defined in TEXTLIB(THASROAR)
   Identified by rec length = 38
   (All SW_AAR will be set to zero)

2 New format fixed length record as defined in TEXTLIB(THASRAAR)
   Identified by rec length = 56

3 CSV file.
   Identified by DSN ending in 'CSV'.
   Field 1: CLASS_SET
   Field 2: zero to 6 blank-separated landmark codes.
   Field 3,4,5: AAR, RW_AAR, SW_AAR.

<u>Locations</u>

The Locations are defined by specifying Landmark Types. Any Landmark types can be specified, but is is intersections which are usually of interest.

The data for Locations average accident rates files are obtained by running the Rate Table Program once for each landmark type (or set of landmark types).

To specify locations which are not at any landmark, code a landmark of 'bl'
(lower case BL).

Locations Average Accident Rate files are used by program THAS210 (Accident Prone Locations).

Sections

The Sections Average Rate files have the same format as Locations files, but the Landmark Type fields are left blank.

The data for this file can be obtained by running the Rate Table Program once, specifying the Highway Classification Sets desired.

Sections Average Accident Rate files are used by program THAS220 (Accident Prone Sections).

PL/I Record Definitions

```
Old format PL/I record definition, THASP.TEXTLIB(THASROAR) :

    DCL AAR_REC CHAR(38);              /* Average_Accident_Rate record */
    DCL 1 AAR DEF AAR_REC,
        5 CLASS_SET CHAR(8),           /* highway classification set   */
        5 LMK_TYPE(6),                 /* up to 6 lmk type codes       */
          8 FILL    CHAR(1),
          8 CODE    CHAR(2),
        5 FILL2     CHAR(1),
        5 AAR       PIC'Z9V.99',       /* average accident rate        */
        5 FILL3     CHAR(1),
        5 RW_AAR    PIC'Z9V.99';       /* road-weighted ave. acc. rate */

New format PL/I record definition, THASP.TEXTLIB(THASRAAR) :


 DCL AAR_REC                CHAR(56); /* Average_Accident_Rate record*/

 DCL 1 AAR                  BASED( ADDR( AAR_REC )),
         5 CLASS_SET        CHAR(20),       /* Hwy classification set*/
         5 LMK_TYPE(6),                     /* Up to 6 lmk type codes*/
           8 X1             CHAR(1),
           8 CODE           CHAR(2),        /* 22 25 28 31 34 37     */
         5 X2               CHAR(1),
         5 AAR              PIC'Z9V.99',    /* Average accident rate  40 */
         5 X3               CHAR(1),
         5 RW_AAR           PIC'Z9V.99',    /* Road-weighted AAR      46 */
         5 X4               CHAR(1),
         5 SW_AAR           PIC'Z9V.99';    /* Severity-weighted AAR  52 */
```

5.5 THASP.CASELIST

This file contains a copy of the Case, Date, Segment and Km from THASP.PROV.VALLOC.CURRENT(0). It is created in update job JU40 by PL/I program THAS080. It exists just so that program THAS012 in update job JU01 can check that new accidents are not already in the system.

Sort order: Case, Date

Record length: 27

```
    DCL 1 OUTSTRUC,
        8 ACCASE    CHAR(8),
        8 C1        CHAR(1),
        8 ACCDATE   CHAR(8),
        8 C2        CHAR(1),
        8 SEGNUM    CHAR(4),
        8 C3        CHAR(1),
        8 KMMARK    CHAR(4);
```

5.6 THASP.CFRANK.FOR140

This file contains a table for converting the character CONTRBnn variables from the accident record into the integer contributing factor variables used in calculating the Road, Driver and Vehicle weights, in program THAS140, in Update job U30.

Sort order: none.

Record Format:

```
  cols 1-2  contributing factor code from the accident record
  cols 4-5  contributing factor code used in program THAS140.
```

5.7 THASP.CLASS.NAMES.scheme

These files defines the Categories and Characteristic Codes which make up the system of highway classifications. See User's Manual section 2.4 for more details on highway classifications.

The Class Names files define the Categories and Characteristics of a highway classification scheme. The scheme name (at the end of the dataset name) must match the scheme name in a corresponding THASP.SEGCLASS.scheme file.

The schemes currently defined are YR1987 - the old 2-category scheme, and YR2002 -the new 6-category scheme.

These files are used by the HAS system (ISPF and PL/I) to dynamically label highway classification fields depending upon the classification scheme which is in effect.

There is a limit of 8 categories, and of 10 characteristics within each category.

The files are column dependent file as in the following (YR1987) example.

```
 DSN=THASP.CLASS.NAMES
----+----1----+----2----+----3----+----4
1 Location                              - Category
  U Urban                               \
  R Rural                                > Characteristics
  X U/U (for unnumbered routes)         /
2 Class                                 - Category
  C Conventional                        \
  E Expressway                           \
  F Freeway                               > Characteristics
  N unNumbered route (900 series)        /
```

Column 1, if non-blank, is a number n indicating that a Category name follows. The number n is that Category's position. E.g. "2 Class" indicates that the second byte of the Classification field is used to describe the highway Class.

If column 1 is blank, then column 3 must contain a one byte Characteristic Code, and columns 5 to 30 must contain the full name of the characteristic.

Category and Class names may be up to 30 characters long.

5.8 THASP.COUNTER.LOCNS - Counter Locations File

This file contains the LKI Locations of traffic volume counters. (Counters not on the LKI are not included.)

```
Sort order   : Station ID.
Record Length: 21

  Columns    Description

   1- 8      Station ID
  10         Station type (P or S)
  12-15      Segment number
  17-21      KMMARK
```

The PL/I record definition for the Counter Locations file follows.

```
  DATASET:    THASP.TEXTLIB
  MEMBER:     THASRTVL

----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+

          /* Traffic Volume Counter Location record: */
          5 STATION       CHAR (8),
          5 BLANK1        CHAR (1),
          5 TYPE          CHAR (1),        /* 'P' for Permanent,
                                              'S' for Short Count */
          5 BLANK2        CHAR (1),
          5 SEGMENT       CHAR (4),
          5 BLANK3        CHAR (1),
          5 KMMARK        PIC 'ZZ9V.9';
```

This file is currently generated on the development workstation, in folder C:\HAS\VOLUME, from MS-Access database VOLUME.MDB, tables:

```
      Counter_Locations             - all individual counter locations
      Counter_Locations_Combined    - generated combined counter locations
                                      for 2-way segments.
```

Using macro ExportCounter_LocationsTable, these are exported into text files:

```
      Counter_Locns_1way.txt
      Counter_Locns_2way.txt
```

These are then combined and sorted into `locnbothways.txt` and uploaded to the mainframe, using `UPLOCNS.BAT`.

5.9 THASP.COUNTER.MAP.INTERSEC.CSV

This file specifies how to combine traffic volume data from count stations to obtain volumes for intersections.

The file is created in Volume utility job THAS760, Proc THASMAP, from regional counter map files.

This file is identical in structure to THASP.COUNTER.MAP.SUBSEG.CSV.

In most cases the START_KM and END_KM will be the same, for each intersection. However END_KM may be greater than START_KM if desired, e.g. for large intersections greater than 100 metres in length.

5.10 THASP.COUNTER.MAP.NODE.CSV

This file specifies how to combine traffic volume data from count stations to obtain traffic volumes for nodes.

The file is created in Volume utility job THAS760, Proc THASMAP, from regional counter map files.

The records are variable length, and contain comma-separated-values. The first 5 fields (22 bytes) are required to be in fixed columns, however, to allow the file to be sorted.

The data is maintained by Regional personnel, in Excel files. See the section on Traffic Volume file management for details.

```
Sort Order:    Node, From_Year

Fields:                          Fixed Columns
      1     Region                    1
      2     Node                      3-10
      3     From Year                12-15
      4     To Year                  13-16
      5     StationID 1
      6     Factor 1
Up to 7 more Station ID and Factor pairs may be specified.
```

SAMPLE

```
5,15051510,1988,2010,48-005C,1,48-007C,1
5,15401550,1988,2010,46-011C,1.5,46-002C,0.5
```

Notes:

- The TO_YEAR field may be blank, if the counter map applies to only one year.
- The year ranges for one node must not overlap.
- See the notes on station IDs in the THASP.COUNTER.MAP.SUBSEG.CSV section below.

5.11 THASP.COUNTER.MAP.SUBSEG.CSV

This file specifies how to combine traffic volume data from count stations to obtain volumes for sub-segments.

The file is created in Volume utility job THAS760, Proc THASMAP, from regional counter map files.

The records are variable length, and contain comma-separated-values. The first 5 fields (22 bytes) are required to be in fixed columns, however, to allow the file to be sorted.

The data is maintained by Regional personnel, in Excel files. See the section on Traffic Volume file management for details.

```
Sort Order:    Segment, From-Year, To-Year, Start_km

Fields:                          Fixed Columns
      1     Region                    1
      2     Segment                   3- 6
      3     From Year                 8-11
      4     To Year                  13-16
      5     Start Km                 18-22   (format ##9.9)
      6     End Km
      7     StationID 1
      8     Factor 1
Up to 7 more Station ID and Factor pairs may be specified.
```

Being variable length, and not completely column-dependent, there is no include file for defining the record. However include file THASDSMF defines the data obtained from each record:

```
          5 SEGMENT       CHAR(4),                    3 -  6
          5 FROM_YEAR     CHAR(4),
          5 TO_YEAR       CHAR(4),                   13 - 16
          5 START_KM      FIXED DEC(4,1),            18 - 22
          5 END_KM        FIXED DEC(4,1),
          5 #STATIONS     FIXED BIN(31),
          5 STATION(8),
            8 ID          CHAR(8),
            8 FACTOR      FLOAT BIN;
```

SAMPLE

```
2,0960,1987,2010, 29.0, 70.0,P-22-1C,1
2,0960,1987,2010, 70.1, 71.1,38-001C,1
2,1105,1987,2010,  0.0,  4.4,P-26-3C,1
2,1110,1987,1990,  0.0, 21.5,26-012C,1
2,1110,1991,1991,  0.0, 21.5,P-26-2C,1.15
2,1110,1992,1992,  0.0, 21.5,26-012C,1
2,1110,1993,1993,  0.0, 21.5,P-26-2C,1.15
2,1110,1994,1998,  0.0, 21.5,26-012C,1
2,1110,1987,2010, 21.6, 39.6,P-26-2C,1
2,1110,1987,1990, 39.7, 47.1,26-009C,1
2,1110,1991,1991, 39.7, 47.1,P-26-2C,1.3
```

Notes:

- each segment may be divided up differently for each year range.
- this file does not have to cover the entire highway network, e.g. entire segments, or parts of segments may be omitted
- year ranges for each segment must not overlap, and must be coded in ascending order.
- in each set of map records following a header record, km ranges must not overlap. I.e. START_KM must be greater than the previous END_KM.
- Counter map files may contain Station ID's which are NOT on the LKI. The Counter_Locations and Counter_Locations_Combined tables of the VOLUME.MDB database describe all the valid Station IDs. A copy of these tables may be in the Counter_Locations.xls Excel file. Note that file THASP.COUNTER.LOCNS contains only stations which are on the LKI.

5.12 THASP.CRITICAL.RATES.scheme

The Accident-Prone Locations and Accident-Prone Sections programs have options to select Accident-Prone sites by a critical Number, Accident Rate and Accident Severity Ratio (ASR). Instead of specifying the critical numbers, the user may specify that they be looked up, using the Traffic Volume and Highway Classification. This is the "Table Lookup" option. THAS210 and THAS220 look for the critical values in this file.

The final part of the dataset name - "scheme" - must be one of the current Highway Classification scheme names. (See the file description of THASP.CLASS.NAMES.scheme) The Highway Classification is a primary field in this file, so it follows that there must be one of these files for each classification scheme.

The file for the original 2-category classification scheme is THASP.CRITICAL.RATES.YR1987. The data in this file was compiled before the "Calculated Critical Rate" option was added to the Accident-Prone Locations and Sections programs.

The file for the new 6-category classification scheme is THASP.CRITICAL.RATES.YR2002. There is no data other than one dummy record in this file (as of April 2002). This file requires actual Highway Classifications, (as opposed to Classification Sets, as in the Average Accident Rates file), so with 6 categories, there are a large number of possible Highway Classifications. Thus using this method of specifying critical values may be impractical for the new highway classification scheme.

There is just one, system-wide Critical Rates file (for each classification scheme) These files can only be modified by privileged users.

The file is allocated with 80 byte records. Text beyond column 60 is ignored.

There is no record definition include file. The file, with only one of its two types of critical rates, is loaded into the Critical Rate Table by routine THASLCR. Routine THASPCR prints the table as read, and THASGCR does a look-up.

Record definition:

Columns	Description
1- 8	Highway Classification
10-16	Minimum ADT in range
18-24	Maximum ADT in range
27-29	Accident-Prone Locations Critical Number
31-35	Critical Rate
37-41	Critical ASR
44-47	Accident-Prone Sections Critical Number
49-53	Critical Rate
55-60	Critical ASR

Sample portion of a Critical Rates file:

```
                              Locations         Sections
Class      ADT-1   ADT-2  Num  Rate  ASR    Num  Rate  ASR
AAAAA    9999999 9999999 9999 99.99 99.999 9999 99.99 99.999
----+----1----+----2----+----3----+----4----+----5----+----6
UC             0    5000    1  2.02  1.01   101  2.62  2.01
UC          5001   10000    2  1.39  1.02   102  1.53  2.02
UC         10001   15000    3  1.21  1.03   103  1.70  2.03
UC         15001   20000    4  1.22  1.04   104  1.39  2.04
UC         20001 9999999    5  1.00  1.05   105  1.27  2.05
UE             0    5000    6  3.23  1.06   106  3.78  2.06
UE          5001   10000    7  1.83  1.07   107  2.44  2.07
```

5.13 THASP.DATASEL - (PDS) - Data Selection Specifications

A Data Selection Specification is the saved contents of the Data Selection Panel (P200) and its sub-panels. There is one saved Data Selection Specification per member of THASP.DATASEL.

DATASEL members are written and read by REXX program THASP200. See that program for a description of the format of the file. (DATASEL members are in the same format as PL/I program THAS200's SYSIN file.)

5.14 <userid>.DATASEL.CURRENT/PREV

Data selection specification files called <userid>.DATASEL.CURRENT and <userid>.DATASEL.PREV are automatically maintained for each user of the H.A.S. system. The formats of the files are the same as for THASP.DATASEL members.

These files are maintained so that the user can always recall his/her last and previous data selection parameters.

See REXX program THASP200, and Panel THASP200.

5.15 THASP.DFSEL - (PDS) - Data Field Selection Specifications

A Data Field Selection Specification is the saved contents of the Data Field Selection (P203) sub-panels. There is one saved Data Field Selection Specification per member of THASP.DATASEL.

DFSEL members are written and read by REXX program THASP203. See that program for a description of the format of the file. (DFSEL members are in the same format as PL/I program THAS203's SYSIN file.)

5.16 THASP.DEFAULTS

5.16.1 Job Priority and Time (PRTYTIME)

File THASP.DEFAULTS(PRTYTIME) contains default priorities and times for each of the Update jobs, and for Data Retrieval jobs. It is read by REXX program THASPTAB. There must be one record for each Update job, and one record to specify the defaults for all Data Retrieval jobs.

Each record in the file must contain three fields, separated by spaces. There is no column dependence.

```
  field 1    - job name, e.g. U80 for update jobs, DRET for Data Retrieval
  field 2    - priority, e.g. 6 for overnight jobs
             - goes on the PRTY= field of the job statement
  field 3     - maximum job CPU time
             - goes on the TIME= field of the job statement

   DSN=THASP.DEFAULTS(PRTYTIME)
----+----1----+----2----+----3----+----4----+----5----+----6----+----7
U00       7    (0,30)
U01       6    10
U10       6    10
U15       6    10
U20       6    10
U25       6    10
U30       6    10
U40       6    10
U45       6    10
U80       6    10
U90       7    (0,30)
DRET      6     5
PDSM      6    15
760       6     1
770       7     1
```

5.16.2 Accident-Prone Locations Defaults (DEF210)

This file contains the default values for the Accident-Prone Locations ISPF panels. See the REXX program and panel THASP210. Note that the report field descriptions are copied from this Defaults file onto the menu.

Note that the control parameter field labels are used to find the default values, so they should not be changed.

In the Reports section, only the SEARCH SEQUENCE label is used to position the file for reading. The others may be changed to change the field labels on the menu. The order of the labels should not be changed.

```
 DSN=THASP.DEFAULTS(DEF210)
 ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
       ===== DEFAULTS FOR THAS210 - HAZARDOUS LOCATIONS ======
                     (Do not change labels on right)

0.0                  LOCATION RADIUS
A1 A2 A3             LANDMARKS (UP TO 6, 2 DIGITS EACH, SEP BY 1 BLANK
01                   LOCATION TYPES (UP TO 6, 2 DIGITS EACH, SEP BY 1 BL)
BYGROUPL             SAS PROGRAM
Y                    LANDMARK DESCRIPTIONS IN REPORTS
.1                   CR_SIG_LEV - Lev of Sig for calculating crit. rates
2.5                  CM_SIG_LEV - Lev of Sig. for Counter-measure method

                     ====== REPORTS =======

                     REGION/DISTRICT PRIMARY SORT (blank, R, D or RD)
                     REGION/DISTRICT PAGE BREAK (blank, R or D)

    In Column 1: Blank for no report, A for Ascending, D for Descending

        1  SEARCH SEQUENCE
        2  Hwy-Segment-Km
        3  Highway Class
        4  Average Daily Traffic (ADT)
        5  Accident Rate
        6  Road-Weighted Acc. Rate
        7  Rate/Critical Rate Ratio
        8  Road-Weighted Rate Ratio
        9  Number of Accidents
       10  Number of Vehicles
       11  Number of Fatal Accidents
       12  Number of Fatalities
       13  No. of Injury Accidents
       14  Number of PDO Accidents
       15  Accident Severity Ratio
------------------------------------------------------------------------------
```

5.16.3 Accident-Prone Sections Defaults (DEF220)

This file contains the default values for the Accident-Prone Sections ISPF panels. See the REXX EXEC program and panel THASP220. Note that the report field descriptions are also copied from this Defaults file onto the menu.

Note that the control parameter field labels are used to find the default values, so they should not be changed.

In the Reports section, only the SEARCH SEQUENCE label is used to position the file for reading. The others may be changed to change the field labels on the menu. The order of the labels should not be changed.

```
 DSN=THASP.DEFAULTS(DEF220)
 ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
       ===== DEFAULTS FOR THAS220 - HAZARDOUS SECTIONS ======
                    (Do not change labels on right)

1.0                 DEFAULT SECTION LENGTH
BYGROUPS            SAS PROGRAM
Y                   LANDMARK DESCRIPTIONS IN REPORTS
                    WORST SECTIONS
.1                  CR_SIG_LEV - Lev of Sig for calculating crit. rates
2.5                 CM_SIG_LEV - Lev of Sig. for Counter-measure method

                    ====== REPORTS =======

                    REGION/DISTRICT PRIMARY SORT (blank, R, D or RD)
                    REGION/DISTRICT PAGE BREAK (blank, R or D)

    In Column 1: Blank for no report, A for Ascending, D for Descending

        1  SEARCH SEQUENCE
        2  Hwy-Segment-Km
        3  Section Length
        4  Highway Class
        5  Average Daily Traffic (ADT)
        6  Accident Rate
        7  Road-Weighted Acc. Rate
        8  Rate/Critical Rate Ratio
        9  Road-Weighted Rate Ratio
       10  Number of Accidents
       11  Number of Vehicles
       12  Number of Fatal Accidents
       13  Number of Fatalities
       14  No. of Injury Accidents
       15  Number of PDO Accidents
       16  Accident Severity Ratio
       17  ACCIDENT RATE IN MOST HAZARDOUS SECTION
------------------------------------------------------------------------------
```

5.16.4 Default Accident Type Costs (COST)

These are the default cost values which were used on the panels for both accident-prone locations and sections. The costs were used to calculate the total cost of accidents at accident-prone sites.

Since the COST column has been removed from the THAS210 and THAS220 reports, this file is no longer used.

```
DSN=THASP.DEFAULTS(COSTS)
      ----+----1----+----2----+----3----+----4----+----5----+----6----+----7
***** ****************************** TOP OF DATA ************************
00000 === DEFAULT COST VALUES FOR HAZARDOUS LOCATIONS AND SECTIONS PROGRAMS ===
00000                    (DO NOT MAKE CHANGES PAST COLUMN 20)
00000
00000 500000             FATAL ACCIDENT COST
00000 20000              INJURY ACCIDENT COST
00000 2000               PDO ACCIDENT COST
00000 0                  ADMIN COST
***** ****************************** END OF DATA ************************
```

5.16.5 Traffic Volume Defaults (TRAFFVOL)

This file contains two values on separate lines, not column-dependent. It is used by program THAS760, which creates the Segment Volumes file.

```
  DSN=THASP.DEFAULTS(TRAFFVOL)
----+----1----+----2----+----3----+----4----+----5----+----6----+----7
50.0              MAX_COUNTER_DISTANCE  (KM.)
50                MAX_PERCENT_VARIANCE
```

The two values are used as follows:

- When a segment has no traffic volume counters, a volume may still be determined for that segment if there is a counter on one or both adjacent segments within MAX_COUNTER_DISTANCE (in kilometers).
- If the annual ADT's for a segment differ by more than MAX_PERCENT_VARIANCE, the segment is marked with asterisks on THAS760's report.

*5.17 THASP.DIAGNOS.**

Files named: THASP.DIAGNOS.LOCATION.DATA
THASP.DIAGNOS.LOCATION.DESC
THASP.DIAGNOS.SECTION.DATA
THASP.DIAGNOS.SECTION.DESC

can be produced by procedure THASWDF, in programs THAS210 and THAS220. The files contain detailed counter-measure data.

The code to produce these files was written for Tarek Sayed, in 1994, for downloading to his PC for further processing in his "Expert System". This feature, as far as I know, was never used by anyone else.

5.18 THASP.DISTRICT

This file associates a region number and a district name with each district number.

The data is maintained, and exported from, the **District** table in the HASLKI Access database.

Sort order: District number
Record Length: 23

Columns	Description
1- 2	District Number, right justified, blank filled
4	Region Number
6-23	District Name

DATASET: THASP.TEXTLIB
MEMBER: TLKIDST

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+

                                        /***** District File *****/
   DCL DST_RECD                  CHAR(23) STATIC ;
   DCL 1 DSTA                    DEF DST_RECD,  /** Alpha Version **/
         5 HWY_DST_NUM           CHAR(02),      /* District-Number */
         5 FILL01                CHAR(01),
         5 HWY_RGN_NUM           CHAR(01),      /* Region-Number   */
         5 FILL02                CHAR(01),
         5 HWY_DST_NAM           CHAR(18);      /* District-Name   */
   DCL 1 DSTN                    DEF DST_RECD,  /** Numeric Ver.  **/
         5 HWY_DST_NUM           PIC'99',       /* District-Number */
         5 FILL01                CHAR(01),
         5 HWY_RGN_NUM           PIC'9',        /* Region-Number   */
         5 FILL02                CHAR(01),
         5 HWY_DST_NAM           CHAR(18);      /* District-Name   */
```

5.19 THASP.FIELD210,220,225

These files define fields available for selection for inclusion in output CSV files.
Each program which has selectable CSV output fields has a dedicated file, eg THASP.FIELD210 for program THAS210.

Each record of the file defines one output field, with:

Field Name	- may contain spaces - maximum length 20 - stored in external defined in THASXAFC. - must correspond with same names hard-coded in PL/I routines 225, CHA and WCS
Colon	
Description	- used only for display on panel THASSSOF.

For details, see REXX program THASSSOF, which reads these files and displays their contents.

Each user has corresponding <userid>.FIELD* files, which contain just the most recently selected field names.

5.20 THASP.FILEINFO

THASP.FILEINFO is a text file containing documentation on all the Highway Network and Parameter files which can be selected for editing in the H.A.S. ISPF Dialog. It is read by REXX program THASEDFL.

The beginning of documentation for a file is indicated by a record starting with three asterisks, followed by a blank, followed by the file (dataset) name. The file name does not include the high level index (e.g. THASP).

All files listed in THASP.TABLE(ELPARAM) and (ELNETDEF) should be included.

The documentation in THASP.FILEINFO should be kept consistent with the documentation in the System Manual.

5.21 THASP.FIXLOG

This is a log of Tape Master file records fixed using program THAS720.

Here are the first few records of the file:

```
*** THAS720: 19940216145001118
'R0324648' '*' 'LOCN_CODE      ' ' 16 15100974'
'R0708664' '*' 'LOCN_CODE      ' '101 25700563'
'R0565614' '*' 'LOCN_CODE      ' '101 25700029'
'R0471083' '*' 'TOTALINJ       ' '000'
'R0729265' '1' 'CONTRB11       ' '00'
*** THAS720: 19940311085739133
'R0084969' '*' 'LOCN_CODE      ' ' 97 11157240'
'R0550282' '*' 'ACCDATE        ' '920516'
*** THAS720: 19940428121106309
'R0084969' '*' 'LOCN_CODE      ' ' 97 11150724'
'R0273163' '*' 'LOCN_CODE      ' '  1 05500563'
'R0075526' '*' 'HIGHLET        ' 'A'
```

The '***' records contain the program identification and a time stamp. The following records show what fields were changed, and what new values were inserted. The second field is the page number: * for all pages.

5.22 THASP.HIGHWAY

This file associates a highway name with each highway.

The data is maintained and exported from the **Highway** table in the HASLKI Access database.

```
Sort order:         Highway Number, Highway Letter
Record length:      65

Record Definition:
```

Columns	Description
1- 3	Highway Number, right justified, blank filled
4	Highway Letter
6-65	Highway Name

```
  DATASET:   THASP.TEXTLIB
  MEMBER:    TLKIHWY
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+
                                          /***** Highway File *****/
    DCL HWY_RECD                  CHAR(65) STATIC ;
    DCL 1 HWY                     DEF HWY_RECD,
          5 HWY_NUM               CHAR(04),       /* Highway-Number */
          5 FILL01                CHAR(01),
          5 HWY_NAM               CHAR(60);       /* Highway-Name   */
```

5.23 THASP.INTERSEC.VOLUMES

This file contains traffic volumes for all sub-segments specified in the THASP.COUNTER.MAP.INTERSEC.CSV file. It is created from the THASP.STATION.VOLUMES and THASP.INTERSEC.COUNT.MAP files, by program THAS762, in traffic volume utility job 761.

This file is identical in structure to THASP.SUBSEG.VOLUMES.

See THASP.SUBSEG.VOLUMES for sort order and record structure details.

5.24 THASP.LABELLED.FOR140

This file contains 994 "labelled observations", for use by program THAS140, in Update job U30, in assigning Driver, Vehicle and Road weights to accidents.

The records are free-form, and must contain 23 real numbers. The first 20 are an "observation" (derived from accident record contributing factors), and the last 3 are the "label": the Driver, Vehicle and Road weights.

This file was created by Tarek Sayed and Walid Abdelwahab (temporary Highway Safety Branch personnel) in 1994. The "observations" were obtained from actual accident records, then the weights were assigned manually (I believe!).

See the description of program THAS140, and/or the THAS140 PL/I program itself for details.

5.25 THASP.LANDMARK

This file contains one record for each landmark on all LKI segments.

Sort order: Segment, KMMARK, LM_NUMBER

Record Definition:

Columns	Description
1 - 4	Segment number, zero filled
6 - 12	Kilometre Offset, decimal point in column 9, two decimal places
14	Landmark Number, for when there are > 1 at same Km.
16 - 17	Landmark Type, two characters, no blanks.
20 - 99	Landmark Description, text.

DATASET: THASP.TEXTLIB
MEMBER: TLKILMK

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+
                                      /***** Landmark File *****/
   DCL LMK_RECD                  CHAR(99) STATIC ;
   DCL 1 LMKA                    DEF LMK_RECD,  /** Alpha Version **/
         5 SEG_NUM               CHAR(04),      /* Segment-Number  */
         5 FILL01                CHAR(01),
         5 KMOSSL                CHAR(07),      /* Kilometre Offset*/
         5 FILL02                CHAR(01),
         5 LM_NUM                CHAR(01),      /* Landmark-Number */
         5 FILL03                CHAR(01),
         5 LM_TYPE               CHAR(02),      /* Landmark-Type   */
         5 FILL04                CHAR(02),
         5 LM_DESC               CHAR(80);      /* Landmark-Desc.  */
   DCL 1 LMKN                    DEF LMK_RECD,  /** Numeric Ver.  **/
         5 SEG_NUM               PIC'9999',     /* Segment-Number  */
         5 FILL01                CHAR(01),
         5 KMOSSL                PIC'ZZZ9V.99', /* Kilometre Offset*/
         5 FILL02                CHAR(01),
         5 LM_NUM                PIC'9',        /* Landmark-Number */
         5 FILL03                CHAR(01),
         5 LM_TYPE               CHAR(02),      /* Landmark-Type   */
         5 FILL04                CHAR(02),
         5 LM_DESC               CHAR(80);      /* Landmark-Desc.  */
```

*5.26 THASP.LANDMARK.MATCH.LIST.**

Landmark Match List files define the transformation from one version of the LKI (or part of the LKI) to the next version. Each Landmark Match List file must have a corresponding Segment Index file. These files are used by PL/I programs THAS105 and THAS106.

See the LKI Conversions section in this manual for details on LKI conversions.

Each Landmark Match List contains an effective-date range line, followed by many lines which match the same landmarks in the two LKI versions. Blank lines may also be present.

Each file may contain match data for one or more date ranges. Multiple match list files are concatinated at run time to be treated as one.

Current Landmark Match List files are:

JAN90	complete 1980-to-1990 transformation.
INCREM	incremental transformations between 1-Jan-1990 and 1-Apr-1995
APR95	complete 1990 to 1995 transformation
JAN00	transformation to 1-Jan-2000: mostly Vancouver Island
APR00	Vancouver Island corrections
MAR01	removal of parts of highway 99A in Surrey.
JAN02	moved start of seg 2348 (Hwy19) to south end from middle of Comox Valley OP
JAN03	deletes Seg 2342 (Hwy 28) as it overlapped with East part of Seg 2341.
MAR03	East half of Seg 0925 (hwy 1 E of Kamloops) becomes split 2080 & 2085
JAN04	Revisions: Hwy1 near Victoria, Nelson area.
APR04	Compressed last 400m of Segment 1501 at Skidegate
JUN04	Highway 14 revision: old seg 0309 now (0370),0371,0372

See the skeleton for job PDSM to see which programs use these files.

Effective-date lines

An effective-date range line consists of two dates in the format YYYYMMDD followed by a label. The label must contain the word 'date' (upper or lower case) to identify it as an effective-date line; otherwise the label is treated as a comment.

Accidents occurring between the two dates (inclusive) will be put through the transformation specified in the Landmark Match List.

```
cols.  1-8    - First effective date
cols. 10-17   -  Last effective date
cols. 18-25   -  Label
```

The effective-date range in any Landmark Match List must be identical to the effective-date range in the corresponding Segment Index. Therefore, there must be only one Landmark Match List (and Segment Index) with a particular date range, although there could conceivably be several with the same start-date and different end-dates, or vice versa.

Data lines

The Landmark Match List File contains one record for each matched landmark. It also contains a record for one or more unmatched landmarks at the beginning or end of a segment. See the LKI Conversion section for details.
??? make sure detailed match list rules are specified here or in the LKI conversion section

```
  cols.  1-4  -   Old segment number.
  cols.  6-11 -   Old kmmark (with two decimal places).
  col.  13    -   Obsolete location marker. "!"
  cols. 15-18 -   New segment number (usually the same as the old segment
                  number).
  cols. 20-25 -   New kmmark (with two decimal places).
```

The PL/I record definitions follow:

```
  DATASET:   THASP.TEXTLIB
  MEMBER:    THASRLIP
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+
    DCL   LMK_MATCH_LINE               CHAR (25);
    DCL 1 LMK_MATCH_INPUT_LINE         DEFINED LMK_MATCH_LINE,
          5 OLD_SEGMENT#     CHAR (4),
          5 SPACE1           CHAR (1),
          5 OLD_KMMARK       PIC 'ZZ9V.99',
          5 SPACE2           CHAR (1),
          5 OBSOLETE_LOCN    CHAR (1),
          5 SPACE3           CHAR (1),
          5 NEW_SEGMENT#     CHAR (4),
          5 SPACE4           CHAR (1),
          5 NEW_KMMARK       PIC 'ZZ9V.99';
    DCL 1 LMK_MATCH_DATE_LINE          DEFINED LMK_MATCH_LINE,
          5 EFFECTIVE_DATE1  CHAR (8),     /* YYYYMMDD */
          5 SPACE            CHAR (1),
          5 EFFECTIVE_DATE2  CHAR (8),
          5 LABEL            CHAR (8);

    DCL   OLD_KMMARK_CHAR              CHAR (6)
                                       DEFINED LMK_MATCH_LINE  POS (6);
    DCL   NEW_KMMARK_CHAR              CHAR (6)
                                       DEFINED LMK_MATCH_LINE  POS (20);



  DATASET:   THASP.TEXTLIB
  MEMBER:    THASXLML
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+
    /* Match list array size must be the same as MAX_LANDMARKS. */
    DCL   MAX_LANDMARKS          FIXED BIN (31)   INIT (12000);

    DCL 1 LANDMARK_MATCH_LIST (12000),
        %INCLUDE THASRLML;



  DATASET:   THASP.TEXTLIB
  MEMBER:    THASRLML
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+
          5 OLD_SEGMENT#    CHAR (4),
          5 OLD_KMMARK      PIC 'ZZ9V.9',
          5 OBSOLETE_LOCN   CHAR (1),
          5 NEW_SEGMENT#    CHAR (4),
          5 NEW_KMMARK      PIC 'ZZ9V.9';



  DATASET:   THASP.TEXTLIB
  MEMBER:    THASRLMO
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+
          5 OLD_SEGMENT#   CHAR (4),
```

```
5 OLD_KMMARK     CHAR (5),
5 OBSOLETE_LOCN  CHAR (1),
5 NEW_SEGMENT#   CHAR (4),
5 NEW_KMMARK     CHAR (5);
```

5.27 THASP.LANDMARK.TYPES

This file contains descriptions for each of landmark types which may be used in the THASP.LANDMARK file.

```
Record length: 80
Sort order:    landmark type

Record layout:
```

Columns	Description
1 - 2	landmark type code, may contain any non-blank characters.
4 - 80	description, left justified.

5.28 THASP.LMATCH

The full descriptive name of the LMATCH file is "The Landmark Type / Location Type Match Table".

In the Highway Accident System, the term "Location Type" is used to describe the 2 digit field called 'Accident Location' on the MV104 form.

The table defines Landmark Types at which accident records must have matching Location Types. The LKI and the MV104 accident report form both have 2 digit codes to describe such places as intersections, railway crossings, bridges etc. but different sets of codes are used.

The LMATCH file is used by the Location Code checking program THAS030 in jobs U10, U15, U20 and U25, if the Landmark/Location Type checking option has been selected in the Parameter Menu.
Note: This feature is not used, because it produced too many mismatches!

If the Location Code of the accident matches a landmark location (when rounded to 1 decimal place - 100 metres) then if the landmark type is on the LMATCH file, the LOCN_TYPE on the accident record must equal the corresponding LOCN_TYPE on the LMATCH file.

Record Definition:

Columns	Description
1-2	Landmark Type
3	reserved for possible landmark sub-type
4	blank
5-6	Location Type

For simplicity, the record length is 80.

One record should be coded for each Landmark/Location Type match which is to be enforced.

This file must exist whether or not Landmark/Location Type checking is turned on. The contents of the file are not used, however, if the checking is turned off.

5.29 THASP.LOCTEXT.MASTER

This is a generation data group. The latest generation contains the PLACE, ON and AT location text, where it exists, for the accidents in the THASP.PROV.VALLOC.CURRENT(0) master file. The THASP.LOCTEXT.VSAM file is created from THASP.LOCTEXT.MASTER(0).

Sort Order: Case, Date
Record Length: 126

THASP.TEXTLIB(THASRLTX):

```
DCL LOCTEXT_REC CHAR(126) STATIC;
DCL LOCTEXT_KEY CHAR(16) DEFINED(LOCTEXT_REC);

DCL 1 LTX DEFINED(LOCTEXT_REC),
      5 ACCASE  CHAR(8),
      5 ACCDATE CHAR(8),
      5 PLACE   CHAR(10),
      5 ON      CHAR(50),
      5 AT      CHAR(50);
```

Note that the key is the accident case AND the date, because there are duplicate case numbers in the system.

This file is of the same format as the THASP.UPyyyymm.LOCTEXT.VALLOC files.
In update job U38, the new LOCTEXT data is merged with generation (0), creating generation (+1).

This file was introduced to the system in October 2003. All the PLACE,ON,AT data was queried from TAS at that time. (The next time the accident record is revised, or when the accident data is moved to a relational database, these location text fields should be added to the main accident record.)

Backup:

A copy of generation 1 is saved in THASP.LOCTEXT.MASTER.GEN1.BACKUP.

The LOCTEXT.MASTER could be recreated, if necessary, from THASP.LOCTEXT.MASTER.GEN1.BACKUP merged with all THASP.UAyyyymm.LOCTEXT.VALLOC files after the 200307 update. (Be aware that there are some duplicates in GEN1.BACKUP and the first post-200307 LOCTEXT.VALLOC files.)

Update job JU40 copies THASP.LOCTEXT.MASTER(0) to THASP.UAyyyymm.LOCTEXT.MASTER. These UA files are kept for three years

5.30 THASP.LOCTEXT.VSAM

This VSAM file:

- is created from, and has the same record format as THASP.LOCTEXT.MASTER(0).
- is re-created each time a new generation of THASP.LOCTEXT.MASTER is created, in update job JU38.
- was added to HAS in October 2003-10-17
- is used by PL/I programs THAS240 and THAS251 to add location text to output.

PL/I subroutine THASLLX can be used to query this VSAM file.
Use SC81171.REXX.EXEC(RUNLTX) to query this VSAM file interactively in TSO.

5.31 THASP.NODE.ACDAT

This is the partitioned dataset which contains all the node accident data of the current production "PDS-Master" accident data files. The record layout is described in **Accident Data Record Description** section above.. See also the **PDS-Master** section in the **Master Files** section.

5.32 THASP.NODE.VOLUMES

THASP.NODE.VOLUMES contains traffic volumes for those nodes and years specified in the file THASP.COUNTER.MAP.NODE.CSV

Created by program THAS764 in traffic volume utility job 761.

```
Record Length: 108
Sort Order:    Node, Year

Include file:  THASRNDV

          /* Node Volumes file record. Length 108.  THASRNDV */
          5 NODE              CHAR (8),
          5 blank1            CHAR (1),
          5 YEAR              CHAR (4),
          5 blank2            CHAR (1),
          5 TYPES             CHAR (2),
          5 blank3            CHAR (1),
          5 ANNUAL_ADT        PIC 'ZZZZZZ9',   /* 7 */
          5 MONTHLY_ADT(12)   PIC 'ZZZZZZ9';   /* 12 * 7 = 84 */
```

The TYPES contains 'P ', 'S ' or 'PS' indicating the type(s) of the traffic counters from which the volume data came: Permanent, Short, or both.

5.33 THASP.NODESEG

This file is generated from the SEGMENT file, by program THAS134, in job PDSM. There is one record for each node - the node name is followed by a list of all the segments which connect at that node.

```
Sort Order:    Node
Record Length: 80

The record is defined in include file: RNSG:

        /* NODE-SEGMENT File Record         THASRNSG    */
        5 NODE          CHAR(8),
        5 SEGMENT(8),
          8 fill1       CHAR(1),
          8 NUMBER      CHAR(4),
          8 fill2       CHAR(1),
          8 BEGEND      CHAR(1),        /* B/E Begins/Ends at this node */
          8 fill3       CHAR(1),
          8 CONTIN      CHAR(1);        /* C/D Continuous or Discontinuous */
```

5.34 THASP.NODEVOL

THASP.NODEVOL contains all the data from THASP.NODE.VOLUMES with all other node volumes filled in by averaging data of segments named in the node name, from file THASP.SEGVOL2.

See THASP.NODE.VOLUMES for record structure and sort order details.

5.35 THASP.PDSMAST.DATELIMS - Date Limits File

This file contains the minimum and maximum accident dates of the current version of the PDS Master Files. It is written by program THAS110 in utility job PDSM when the Master Files are recreated. (It is initially called THASP.PDSM.DATELIMS, and is renamed to PDSMAST.DATELIMS when the PDSM files are implemented in job PDSI.)

REXX program THASP200 reads in the date limits, inserts them as defaults in the Data Selection screen, and does not allow dates outside that range to be specified. If either date limit is blanked out by the user, the default is reinserted. Thus explicit date limits go into the JCL, into program THAS200, into subset Description files, and into reports.

Other programs may use the file to obtain default date limits.

Record Definition:

Columns	Description
1- 8	Earliest date in PDS Master Files (YYYYMMDD).
10-17	Latest date in PDS Master Files (YYYYMMDD).
19-26	Maximum date for good statistics (YYYYMMDD) (data after this date is not all in yet) (added Sept 2003)
28-35	Date that Master Files were recreated (YYYYMMDD).
36-44	Time that Master Files were recreated (HHMMSSsss).

5.36 THASP.PROMOTE.LOG

This a sequential file to which the promote utility THASD.REXX.EXEC(PROMOTE) writes module promotion log records.

5.37 THASP.RANGE.FEATURES

The Range Features file defines road-related features, such as guard rails, which extend from a start-location to and end-location.

The intention (in March 1994) was that this file would be loaded with data from the RFI (Road Features Inventory) system, but the project was never completed.

Sort order: Segment, Start-Km

The record format is defined in the following two include files:

```
SAS - THASP.SASUTIL(DESCRGF)
PLI - THASP.TEXTLIB(THASRRGF)
```

Columns	Field
1 - 4	Segment
6 - 11	Start_KM ZZ9V.99
13 - 18	End_KM
20 - 21	District (blank filled, right justified)
22	RFI contract area
24 - 31	Date: for RFI data, the date this record was last modified
33 - 34	Feature Type
36	Side: L, R or C for Left, Right, Centre
38	Functional Class
40	Sub-type
42	EXTRA1
44	EXTRA2
46 - 54	TOTAL
56	Direction: N, S, E, W
58 - 79	Notes

There is one blank at the end of the record to bring the record length to 80.

The contents of many of the fields depend upon the Feature Type. The field contents may also vary from district to district. The feature types and their related data are described on the following pages. The information comes from notes made while examining the RFI data for the RFI to LKI conversion project.

File THASP.RANGE.FEATURES - Feature Types

Type Description and Notes

2 - Functional classes
CLASS: A - arterial; M - main; S - secondary; N - minor; T - trunk; C - collector.

3 - Shoulders
SIDE: R - Right; L - Left; C - Center;
(a Center Shoulder is a median.)
TOTAL: Width of the shoulder, measured in meters.
TYPE: P - paved; G - gravel; T - treated.

4 - Ditches
There are Center Ditches as well as Right and Left ones. They occur on medians.

5 - Dust sites
Highways are generally divided into 6 dust sites per kilometer, and the maintenance department uses them for measuring the amount of dust on the highway.
TYPE: No values are given in the RFI documentation, but the data files have values of L, R, and G.

7 - Guard rails
TOTAL: Installation date.

Some of the dates are 5 digits (e.g. 69679) like the modification dates. However, other installation dates are only 4 digits (e.g. 6538). The reason is that this field originally had only four digits, due to a programming error, and so dates were truncated. There is no way to fix the old data.

TYPE: A - 690mm NP; B - 460mm NP; C - 760mm NP; D - flex beam NTS; E - flex beam TS; F - new 690mm NP; G - new median NP; H - wood; U - uncertain.

The abbreviations NP, NTS, and TS mean "No Posts," "Not To Standard," and "To Standard," respectively.

EXTRA1: No values are given in the RFI documentation, but in some cases the data files have values of N and R.

EXTRA2: No values are given in the documentation, but in some cases the data files have values of G and N.
James Lee says these undocumented values in the EXTRA fields are probably not valid.

10 - Mowing swaths
Mowing swath records seem to appear in pairs, differing only in that SIDE is Left in one record of the pair, and Right in the other record.

16 - Curbs and gutters
TYPE: C - concrete; A - asphalt; S - sidewalk.

17 - Surface types
SIDE: is generally Center.

TOTAL: Pavement year. In the data, the pavement year seems to be

given as two digits only (e.g. 87). It may also be zero.

TYPE: 1 - hot mix; 2 - cold mix; 3 - concrete;
4 - surface treated; 5 - gravel; 6 - dirt; 7 - other;
E - clear; F - unclear.
"Clear" means that the trees have been cleared; "Unclear" means that trees have not been cleared. Both codes indicate a future, rather than an existing, road.

EXTRA1 & EXTRA2:
Both are Seal Coat Year. Putting them together appears to give a two-digit year, e.g. "8" & "8" for 1988, or "9" & "1" for 1991. May also be blank & "0" (for no year known, or for no seal coat?)

19 - Road classes

The road class indicates the frequency of maintenance. Classes 1 and A are the most frequently maintained.

SIDE: is generally Center.
EXTRA1: Summer class:
Values: 1; 2; 3; 4; 5; 6; 7; 8.
EXTRA2: Winter class.
Values: A; B; C; D; E; F.

20 - Profiles

TOTAL: Number of lanes. For RFI-segments covering both sides of the road, this is the number of lanes in the whole road. For RFI-segments covering only one side of the road, this field is the number of lanes on the side belonging to the RFI-segment.

TYPE: D - divided; U - undivided; 3 - 2 lane; 4 - 1 lane;
5 - 3 lane; 6 - exit ramp; 7 - special; 8 - other.

The commonest values, by far, in the data looked at are 3, 4, and 5, giving the number of lanes. The number of lanes given by the code in the TYPE field seems to be identical to the number of lanes in the TOTAL field.

22 - Reference ranges

TYPE: A; B; C; F - fencing; P - pole; R - retaining wall;
S - snow fencing; U - ungulate fencing;
W - other or special.

A, B, and C are different types of fencing.

W is given twice, with the two different meanings ("other" and "special"), in the documentation. N occurs in the data, but is not documented. In some other places in this file, N means "other"; James Lee believes that N means "other" here, too, and W should mean "special."

23 - Rights of way

A right-of-way is a strip of land wider than the road, owned by the highway.

TOTAL: Width, measured in meters.
TYPE: 1 - legal survey; 2 - gazette; 3 - plan; 4 - section 4.

5.38 THASP.RCMP.DETACH

This is an H.A.S. copy of the LKI file TLKIP.RCMP.DETACH.

It is used only by the Details Report program (THAS240) to look up police detachment names.

Sort order: Detachment number.

Record Definition:

Columns	Description
1- 4	Detachment Number (Police Code), numeric, zero filled
6-35	Detachment Name

DATASET: THASP.TEXTLIB
MEMBER: TLKIPDT

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+
                                  /***** RCMP Detachment File *****/
   DCL PDT_RECD                  CHAR(35) STATIC ;
   DCL 1 PDTA                    DEF PDT_RECD,  /** Alpha Version **/
         5 RCMP_DETACH_NUM       CHAR(04),      /* RCMP Detach. No.*/
         5 FILL01                CHAR(01),
         5 RCMP_DETACH_NAM       CHAR(30);      /* RCMP Detach Name*/
   DCL 1 PDTN                    DEF PDT_RECD,  /** Num. Version  **/
         5 RCMP_DETACH_NUM       PIC'9999',     /* RCMP Detach. No.*/
         5 FILL01                CHAR(01),
         5 RCMP_DETACH_NAM       CHAR(30);      /* RCMP Detach Name*/
```

5.39 THASP.REGION

This file associates region names with region numbers.

Sort Order: Region Number
Record Length: 22

```
THASP.TEXTLIB(TLKIREG)
                                          /***** Region File *****/
    DCL REG_RECD                 CHAR(22) STATIC ;
    DCL 1 REGA                   DEF REG_RECD,  /** Alpha Version **/
          5 HWY_REG_NUM          CHAR(01),      /* Region-Number   */
          5 FILL01               CHAR(01),
          5 HWY_REG_NAM          CHAR(20);      /* Region-Name     */
    DCL 1 REGN                   DEF REG_RECD,  /** Numeric Ver.  **/
          5 HWY_REG_NUM          PIC'9',        /* Region-Number   */
          5 FILL01               CHAR(01),
          5 HWY_REG_NAM          CHAR(20);      /* Region-Name     */
```

5.40 THASP.SEG.ACDAT

This is the partitioned dataset which contains all the non-node accident data of the current production "PDS-Master" accident data files. The record layout is described in **Accident Data Record Description** section above.. See also the **PDS-Master** section in the **Master Files** section.

5.41 THASP.SEGCLASS.scheme - Segment Class Files

The SEGCLASS files contain the Highway Classification data for each classification scheme. The classification scheme is defined in the corresponding THASP.CLASS.NAMES.*scheme* file. The currently defined schemes are YR1987 - the old 2-category scheme, and YR2002 - the new 6-category scheme.

Each record defines the highway classification for all or part of one segment, for a date range. The End-KMMARK may be given as 999.9, to indicate that the section runs to the end of the segment.

Important: while it is logically possible to use different sets of date ranges for different parts of a segment, it is recommended that the whole segment be defined with one date range, then defined again with the next date range, even if this means coding the same classification, for the same section, in the two date ranges. (PL/I routine THASGSC would have to be recoded to eliminate this restriction.)

This data is maintained in the SEGCLASS_*scheme* tables of the HASLKI MS-Access database, exported, and uploaded to the mainframe.

```
Sort order: Segment, From_Date, Start_Kmmark.

Record Definition:

      Columns     Description

       1-4        Segment number
       6-13       From_Date - start of date range
      15-22       To_Date - end of date range
      24-30       Start KMMARK (start of the highway section)
      32-38       End KMMARK (end of the highway section)
      40-47       Highway Classification
      48-51       blank (to have same record length as
                  THASP.SEGCLASS.scheme.SORTED)
```

Notes:

- multiple date ranges are allowed for the same section of road. The dates refer to the condition of the road itself, and are independent of segment number changes and effective dates.
- The start and end kms may be coded to 2 decimal places, and the start km may equal the end km of the previous record. End kms may be coded as 999.90 to indicate "end of segment". (PL/I routine THASLSC, which reads this file into memory, rounds to 1 decimal place, separates sections by 100 m, and inserts segment lengths as required.)

After updating this file, if must be sorted by search sequence into file THASP.SEGCLASS.scheme.SORTED. Utility job 770 does the sort - it is submitted automatically if the file is edited using the "Edit Highway Network Definition Files" function of the H.A.S. IPSF Dialog.

The PL/I include files for this dataset are included in the following section.

5.42 THASP.SEGCLASS.scheme.SORTED

These files are sorted versions of the THASP.SEGCLASS.scheme files. They are sorted so that segments occur in the same order as in the Segment-Highway-Node file (SHNFIL), and by KMMARK within each segment.

The SHNFIL sequence number is the index of each segment within the Segment-Highway-Node file, or, equivalently, within the Highway-Segment-Node table (HSNTAB)

```
Sort Order:   HSNTAB sequence number, From_Date, Start_KM
            = Highway, Segment, Segment-Search-Sequence, From_Date, Start_KM

Record Definition:

The PL/I record definitions follow.

  DATASET:   THASP.TEXTLIB
  MEMBER:    THASRSCL

                                  /* SEGCLASS file record: */
          5 SEG_NUM          CHAR (4),                          /*  1- 4 */
          5 X1               CHAR (1),
          5 FROM_DATE        CHAR (8),     /* yyyymmdd */       /*  6-13 */
          5 X2               CHAR (1),
          5 TO_DATE          CHAR (8),                          /* 15-22 */
          5 X3               CHAR (1),
          5 START_KM         PIC 'ZZZ9V.99',                    /* 24-30 */
          5 X4               CHAR (1),
          5 END_KM           PIC 'ZZZ9V.99',                    /* 32-38 */
          5 X5               CHAR (1),
          5 CLASS            CHAR (8),                          /* 40-47 */
          5 HSNTAB_SEQUENCE  PIC 'ZZZ9';                        /* 48-51 */
                                                 /* inserted when sorted */
```

5.43 THASP.SEGDIST - Segment-District File

The SEGDIST file contains a record for each segment or sub-segment which is entirely in one Highway District. Thus this file can be used to look up the District for any Segment-KMMARK.

This file is maintained in the SEGDIST table of the HASLKI MS-Access database.

```
Sort order: Segment, Start_km

Record Definition:
```

Columns	Description
1- 4	Segment Number, zero filled
6-12	Start Km., decimal pt. in col 10, two decimal places
14-19	End Km., decimal pt. in col 17, two decimal places
21-22	District Number

After updating this file, it must be sorted by search sequence into file THASP.SEGDIST.SORTED. Utility job 775 does the sort - it is submitted automatically if the file is edited using the "Edit Highway Network Definition Files" function of the H.A.S. IPSF Dialog.

```
/*---THASRISD---SEGDIST---Segment-District Intersection File -----*/
/* 1999/09/17 MN - record layout changed.                         */

DCL ISD_REC                  CHAR(22) STATIC ;
DCL 1 ISD_A     DEF ISD_REC,
      5 SEGMENT              CHAR(04),      /* Segment Number  */
      5 FILL01               CHAR(01),
      5 START_KM             CHAR(07),      /* Starting Km     */
      5 END_KM               CHAR(07),      /* Ending Km       */
      5 FILL02               CHAR(01),
      5 DISTRICT             CHAR(02);      /* Highway District*/
DCL 1 ISD_N     DEF ISD_REC,
      5 SEGMENT              PIC'9999',     /* Segment Number  */
      5 FILL01               CHAR(01),
      5 START_KM             PIC'ZZZ9V.99',
      5 END_KM               PIC'ZZZ9V.99',
      5 FILL02               CHAR(01),
      5 DISTRICT             PIC'99';       /* Highway District*/
```

5.44 THASP.SEGDIST.SORTED

This is a version of file THASP.SEGDIST with the following modifications:

- sorted in HSNTAB and START_KM order, so that segments occur in Highway, Segment-Search-Sequence order, (to match file THASP.SHNFIL).
- the start and end KMMARKs are formatted consistently (with no leading zeros, decimal point always in the same column) so that sorting is done correctly.

This file should not be updated directly. Update file THASP.SEGDIST, then run Utility job 775 (see THASP.SRCELIB(THAS755)) to recreate THASP.SEGDIST.SORTED.

File THASP.SEGDIST.SORTED is only used at present by program THAS200, but it may be worth converting all programs which use THASP.SEGDIST to use THASP.SEGDIST.SORTED.

The HSNTAB sequence number is the index of each segment within the Highway-Segment-Node table (HSNTAB), or, equivalently, within the SHNFIL.

Record Definition:

Columns	Description
1- 4	Segment Number, zero filled
6-12	Start Km., decimal pt. in col 10, two decimal places
14-19	End Km., decimal pt. in col 17, two decimal places
21-22	District Number
24-26	HSNTAB sequence number (sort key)

```
/*---SEGDIST.SORTED---Segment-District Intersection File record */
/* 1999/09/17 MN - revised                                       */

DCL SDS_REC                  CHAR(26) STATIC;
DCL 1 SDS_A DEF SDS_REC,
      5 SEGMENT              CHAR(04),      /* Segment Number  */
      5 FILL02               CHAR(01),
      5 START_KM             CHAR(07),      /* Starting Km     */
      5 END_KM               CHAR(07),      /* Ending Km       */
      5 FILL01               CHAR(01),
      5 DISTRICT             CHAR(02),      /* Highway District*/
      5 FILL03               CHAR(01),
      5 HSNTAB_SEQUENCE      CHAR(03);

DCL 1 SDS_N DEF SDS_REC,
      5 SEGMENT              PIC'9999',
      5 FILL02               CHAR(01),
      5 START_KM             PIC'ZZZ9.V99',
      5 END_KM               PIC'ZZZ9.V99',
      5 FILL01               CHAR(01),
      5 DISTRICT             PIC'99',
      5 FILL03               CHAR(01),
      5 HSNTAB_SEQUENCE      PIC'ZZ9';
```

```
DATASET:   THASP.TEXTLIB
MEMBER:    THASXSDT
```

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+
    /* ================ Segment-Region-District Table ================ */
    DCL #SDIST                    FIXED BIN (31) EXTERNAL INIT (0);
    DCL MAXSDIST                  FIXED BIN (31)          INIT (800);
    DCL 1 SDISTAB(800)            EXTERNAL,
          5  SEGMENT               CHAR(04) ,
          5  REGION                CHAR(01) ,
          5  DISTRICT              CHAR(02) ,
          5  START_KM              FIXED DEC(4,1) ,
          5  END_KM                FIXED DEC(4,1) ,
          5  DISTRICT_NAME         CHAR(18) ;
```

5.45 THASP.SEGDTCH

This file defines each police detachment's area.
This file is used by program THAS030, if police code vs segment checking is requested, and by program THAS005 in job U05.

```
Sort order: Segment, Detachment

Record Definition:
```

Columns	Description
1- 4	Segment Number, numeric, zero filled
6- 9	Detachment Number (Police Code), numeric, zero filled

```
  DATASET:   THWYP.TEXTLIB
  MEMBER:    TLKIISP
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+
                 /***** Segment-RCMP Detach Intersection File *****/
   DCL ISP_RECD                  CHAR(10) STATIC ;
   DCL 1 ISPA                    DEF ISP_RECD,  /** Alpha Version **/
         5 SEG_NUM               CHAR(04),      /* Segment Number  */
         5 FILL01                CHAR(01),
         5 RCMP_DETACH_NUM       CHAR(04),      /* RCMP Detach Num.*/
         5 FILL02                CHAR(01);
   DCL 1 ISPN                    DEF ISP_RECD,  /** Alpha Version **/
         5 SEG_NUM               PIC'9999',     /* Segment Number  */
         5 FILL01                CHAR(01),
         5 RCMP_DETACH_NUM       PIC'9999',     /* RCMP Detach Num.*/
         5 FILL02                CHAR(01);
```

5.46 THASP.SEGMENT

This is the master segment definition file. It is maintained in the HASLKI MS-Access database.

This file contains the data which were in the old LKI files SEGMENT, SEGWAY, SEGNODE and HWYSEG.

Date formats are YYYYMMDD.

See the **Network Terms and Concepts** section of the **HAS User's Manual** for definitions of Node, Search Sequence, Continuity etc.

Sort order: Segment number.

Record Definition:

Columns	Description
1- 4	Segment Number, numeric, zero filled
6- 8	Highway Number
9	Highway Letter
10 13	Second Highway number and letter (optional)
14 17	Third Highway number and letter (optional)
19 21	Segment Search Sequence
23 25	Segment Report Sequence
27 33	Segment Length, km, 2 dec. pl.
35 42	Segment Effective Date (Date police started using this version)
44 51	Segment Create Date (Date this seg # added to LKI)
53 60	Road Add Date (date road added to LKI, any seg #)
62 69	Begin Node
71	Begin Continuity
73 80	End Node
82	End Continuity
84	Number of ways (1 or 2)
86 89	Opposite Segment
91 140	Segment Description, text

DATASET: THASP.TEXTLIB
MEMBER: THASRSEG

```
                                        /***** Segment File *****/
                    /* Changed to combined segment file Sept. 1999 */
                    /* 2004-12-31: added create and road add dates */
  DCL SEG_RECD                  CHAR(140) STATIC ;
  DCL 1 SEGA                    DEF SEG_RECD, /** Alpha Version **/
        5 SEG_NUM               CHAR(4),     /* Segment-Number  */
        5 FILL01                CHAR(1),
        5 HWY                   CHAR(4),     /* highway            */
        5 HWY2                  CHAR(4),     /* 2nd hwy num+letter */
        5 HWY3                  CHAR(4),     /* 3rd hwy num+letter */
        5 FILL04                CHAR(1),
        5 SEG_SEARCH_SEQ        CHAR(3),
        5 FILL05                CHAR(1),
        5 SEG_REPORT_SEQ        CHAR(3),
        5 FILL06                CHAR(1),
        5 SEG_LEN               CHAR(7),     /* Seg-Length (Km) */
        5 FILL07                CHAR(1),
        5 SEG_EFFECTIVE_DATE    CHAR(8),     /* date police started using
*/
        5 FILL08                CHAR(1),
        5 CREATE_DATE           CHAR(8),   /* date this seg # added to LKI
*/
```

```
          5 FILL08a               CHAR(1),
          5 ROAD_ADD_DATE         CHAR(8), /* dt road added to LKI (any
seg#)*/
          5 FILL08b               CHAR(1),
          5 BEGIN_NODE            CHAR(8),
          5 FILL09                CHAR(1),
          5 BEGIN_CONTINUITY      CHAR(1),      /* C OR D */
          5 FILL10                CHAR(1),
          5 END_NODE              CHAR(8),
          5 FILL11                CHAR(1),
          5 END_CONTINUITY        CHAR(1),
          5 FILL12                CHAR(1),
          5 NWAY                  CHAR(1),      /* 1 OR 2 */
          5 FILL13                CHAR(1),
          5 OPPOSITE_SEG          CHAR(4),
          5 FILL14                CHAR(1),
          5 SEG_NAM               CHAR(50);     /* Segment-Name    */

    DCL 1 SEGN                    DEF SEG_RECD,  /** Numeric Ver.  **/
          5 SEG_NUM               PIC'9999',     /* Segment-Number  */
          5 FILL01                CHAR(01),
          5 HWY,
            10 NUM                PIC'ZZ9',
            10 LET                CHAR(1),
          5 HWY2,
            10 NUM                PIC'ZZ9',
            10 LET                CHAR(1),
          5 HWY3,
            10 NUM                PIC'ZZ9',
            10 LET                CHAR(1),
          5 FILL04                CHAR(1),
          5 SEG_SEARCH_SEQ        PIC'999',
          5 FILL05                CHAR(1),
          5 SEG_REPORT_SEQ        PIC'999',
          5 FILL06                CHAR(1),
          5 SEG_LEN               PIC'ZZZ9V.99', /* Seg-Length (Km) */
          5 FILL07                CHAR(1),
          5 SEG_EFFECTIVE_DATE    PIC'99999999',
          5 FILL08                CHAR(1),
          5 CREATE_DATE           PIC'99999999',
          5 FILL08a               CHAR(1),
          5 ROAD_ADD_DATE         PIC'99999999',
          5 FILL08b               CHAR(1),
          5 BEGIN_NODE            CHAR(8),
          5 FILL09                CHAR(1),
          5 BEGIN_CONTINUITY      CHAR(1),      /* C OR D */
          5 FILL10                CHAR(1),
          5 END_NODE              CHAR(8),
          5 FILL11                CHAR(1),
          5 END_CONTINUITY        CHAR(1),
          5 FILL12                CHAR(1),
          5 NWAY                  PIC'9',       /* 1 OR 2 */
          5 FILL13                CHAR(1),
          5 OPPOSITE_SEG          PIC'9999',
          5 FILL14                CHAR(1),
          5 SEG_NAM               CHAR(50);      /* Segment-Name    */
```

*5.47 THASP.SEGMENT.INDEX.**

There must be a Segment Index file for each Landmark Match List file. (See the THASP.LANDMARK.MATCH.LIST.* section above) Eg THASP.SEGMENT.INDEX.JAN90 is the segment index file for THASP.LANDMARK.MATCH.LIST.JAN90.

A segment index file consists of:

- an Effective Date line (same as in a Landmark Match List file)
- a Report-Missing-Segments line (optional)
- Data Lines: one per segment.

This sequence will be repeated in one segment index file, if the corresponding landmark match list file has more than one Effective Date line.

Blank lines may also be included: they are ignored.

Effective-date lines

Effective date range lines are coded exactly as in the landmark match file. The dates must exactly match those in the corresponding landmark match file: this is how the correspondence of the data in the two files is established. It follows that there must be only one Segment Index (and Landmark Match List) with a particular date range, although there could conceivably be several with the same start-date and different end-dates, or vice versa.

Report-missing-segments lines

A report-missing-segments line consists of a flag that may be either 'YES' or 'NO', followed by a label. The label must contain the word 'report' (upper or lower case) to identify it as a report-missing-segments line; otherwise the label is treated as a comment, and may contain any desired identifying information.

If the flag is specified as 'YES', the Segment Index is a complete-transformation file and should contain all segments in the highway system. Any accidents found with segment numbers not in this Segment Index will be reported as bad locations and will not be saved in the PDS Master Files.

If the flag is specified as 'NO', or if there is no report-missing-segments line, the Segment Index contains only certain segments. It is an incremental-transformation file, or possibly a file for a complete LKI transformation where unchanged segments were deliberately omitted to save time.

```
cols. 1-3   -   Flag
cols. 4-80  -   Label
```

Data lines

The lines for individual segments consist of a segment number and some special-characteristics flags (described in detail in the System Manual section 11.5.3 on "Unchanged, Added and Removed Segments").

```
cols. 1-4   -     Segment number.
col.  6     -     Segment-added flag.
col.  7     -     Segment-removed flag.
col.  8     -     Segment-unchanged flag.
col.  9     -     Segment-has-obsolete-locations flag
```

The PL/I record definitions follow.

```
  DATASET:   THASP.TEXTLIB
  MEMBER:    THASRSIP

----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+

    DCL   SEG_INDEX_LINE                CHAR (80);
    DCL 1 SEG_INDEX_INPUT_LINE          DEFINED SEG_INDEX_LINE,
          5 SEGMENT#                    CHAR (4),
          5 SPACE                       CHAR (1),
          5 SEGMENT_ADDED               CHAR (1),
          5 SEGMENT_REMOVED             CHAR (1),
          5 SEGMENT_UNCHANGED           CHAR (1),
          5 SEGMENT_HAS_OBSOLETE_LOCNS  CHAR (1);
    DCL 1 SEG_INDEX_DATE_LINE           DEFINED SEG_INDEX_LINE,
          5 EFFECTIVE_DATE1             CHAR (8),   /* YYYYMMDD */
          5 SPACE                       CHAR (1),
          5 EFFECTIVE_DATE2             CHAR (8),
          5 LABEL                       CHAR (63);
    DCL 1 SEG_INDEX_RPT_LINE            DEFINED SEG_INDEX_LINE,
          5 REPORT_MISSING_SEGS         CHAR (3),   /* YES OR NO */
          5 LABEL                       CHAR (77);



  DATASET:   THASP.TEXTLIB
  MEMBER:    THASXSNX

----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+

    /* Segment index array size must be the same as MAX_SEGMENTS. */
    DCL   MAX_SEGMENTS           FIXED BIN (31)   INIT (800);

    DCL 1 SEGMENT_INDEX (800),
        %INCLUDE THASRSNX;



  DATASET:   THASP.TEXTLIB
  MEMBER:    THASRSNX

----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+

          5 SEGMENT#                    CHAR (4),
          5 EFFECTIVE_DATE1             CHAR (8),     /* YYYYMMDD */
          5 EFFECTIVE_DATE2             CHAR (8),
          5 FIRST_LMK_REC               PIC 'ZZZZ9',
          5 LAST_LMK_REC                PIC 'ZZZZ9',
          5 SEGMENT_ADDED               CHAR (1),
          5 SEGMENT_REMOVED             CHAR (1),
          5 SEGMENT_UNCHANGED           CHAR (1),
          5 SEGMENT_HAS_OBSOLETE_LOCNS  CHAR (1);
```

5.48 THASP.SEGVOL1

The SEGVOL1 Traffic Volumes file is derived from the Station Volumes file and the Counter Locations file by program THAS760, in job 761.

The file contains an approximation of traffic volumes for segments and years for which volume data is available. A single set of ADTs is derived for an entire segment, although in fact one portion of a segment may have a very different traffic volume from another portion of the same segment.

The accuracy of the data is highly variable:

- data from Short Count stations are approximate.
- traffic volumes are not known for some segments and some years,
- data for some segments are derived from Counters on adjacent segments.

The data in this file becomes the 'default' traffic volume data for files SEGVOL2, SEGVOL3 and NODEVOL. Data from SEGVOL1 is used to fill in the gaps for subsegments and years not specified in THASP.COUNTER.MAP.SUBSEG.CSV, THASP.COUNTER.MAP.INTERSEC.CSV and THASP.COUNTER.MAP.NODE.CSV.

See file THASP.SUBSEG.VOLUMES for record definition and sort order information.

5.49 THASP.SEGVOL2

This file contains the data of file THASP.SEGSEG.VOLUMES, with the gaps (in time and space) filled in with data from THASP.SEGVOL1.

Created by program THAS766 in traffic volume utility job 761.

See file THASP.SUBSEG.VOLUMES for record definition and sort order information.

5.50 THASP.SEGVOL3

This file contains the data of file THASP.INTERSEC.VOLUMES, with volume data for the spaces between the intersections filled in with data from THASP.SEGVOL2.

Created by program THAS766 in traffic volume utility job 761.

See file THASP.SUBSEG.VOLUMES for record definition and sort order information.

5.51 THASP.SHNFIL - Segment-Highway-Node File

This file contains most of the information from the THASP.SEGMENT file, sorted in Highway order.

Prior to an LKI (SEGMENT file) update, the SHNFIL should be copied to save the 'old' version for reference, and for use in the transformation process in job PDSM.
Eg: THASP.SHNFIL.MAR95 - version prior to the April 1995 LKI update
THASP.SHNFIL.DEC99 - version prior to the Jan 2000 LKI update.

SHNFIL has a different record layout to the SEGMENT file for historical reasons: The information now in the SEGMENT file used to be stored in the old files THASP.HWYSEG, SEGMENT, SEGNODE, and SEGWAY. The data was combined into SHNFIL by program THAS132 in job PDSM. THAS132 now simply copies the data from the SEGMENT into the SHNFIL format.

This file should not be changed manually. If the SEGMENT file is changed, job PDSM should be re-run to make sure that those changes are reflected in THASP.SHNFIL, and in the PDS Master.

Note that the Segment Name field (the last one on each record) is not used at present by any H.A.S. programs. However, THAS230 (Histogram Report) uses the equivalent Segment Name field from the SEGMENT file.

```
Sort Order:Highway, Segment-Search-Sequence.


The PL/I definition of the Segment-Highway-Node file follows.

  DATASET:   THASP.TEXTLIB
  MEMBER:    THASRSHN

----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+
    /* changed SEGHN_RECORD length from 107 to 109 *        2kdh    */
    /* 2001-10-10 MN changed seg_search_seq from char(4) to char(3) */
                           /* SEGMENT-HIGHWAY-NODE FILE RECORD    */
    DCL SEGHN_RECORD CHAR(109);
    DCL 1 SEGHN DEF SEGHN_RECORD,
        5 HWY,
          10 NUM             CHAR(3),
          10 LET             CHAR(1),
        5 SEG_SEARCH_SEQ     CHAR(3),
        5 FILL1              CHAR(2),
        5 SEG_NUM            CHAR(4),
        5 FILL2              CHAR(1),
        5 SEG_LEN            PIC'999V.99',
        5 FILL3              CHAR(1),
        5 SEG_EFFECTIVE_DATE CHAR(8),       /* 2kdh */
        5 FILL4              CHAR(1),
        5 BEGIN_NODE         CHAR(8),
        5 FILL5              CHAR(1),
        5 BEGIN_CONTINUITY   CHAR(1),     /* C OR D */
        5 FILL6              CHAR(1),
        5 END_NODE           CHAR(8),
        5 FILL7              CHAR(1),
        5 END_CONTINUITY     CHAR(1),
        5 FILL8              CHAR(1),
        5 NWAY               CHAR(1),     /* 1 OR 2 */
        5 FILL9              CHAR(1),
        5 OPPOSITE_SEG       CHAR(4),
        5 FILL10             CHAR(1),
        5 SEG_NAM            CHAR(50);
```

The PL/I definition of the internal Highway-Segment-Node table follows.

```
  DATASET:   THASP.TEXTLIB
  MEMBER:    THASXSNT

----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+


    /*------------- Highway-Segment-Node Table -----------------------*/

    DCL MAXHSN        EXTERNAL FIXED BIN(31) INIT(500); /* as declared*/
    DCL #HSN          EXTERNAL FIXED BIN(31);           /* # filled   */
    DCL 1 HSNTAB(500) EXTERNAL,
    %INCLUDE THASRSNT;



  DATASET:   THASP.TEXTLIB
  MEMBER:    THASRSNT

----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+

                                      /* SEGMENT-NODE TABLE RECORD */
        5 HWY,
          10 NUM           CHAR(3),
          10 LET           CHAR(1),
        5 SEG_NUM          CHAR(4),
        5 SEG_DATE         CHAR(6),
        5 SEG_LEN          FIXED DEC(4,1),
        5 BEGIN_NODE       CHAR(8),
        5 END_NODE         CHAR(8),
        5 BEGIN_CONT       BIT(1),
        5 END_CONT         BIT(1),
        5 NWAY             CHAR(1),          /* 1 OR 2 */
        5 OPPOSITE_SEG     CHAR(4);
```

5.52 THASP.STATION.LIST

This file contains just one field: an 8 character counter station ID.
It contains a unique, sorted list of all the station IDs in the following files:

```
THASP.COUNTER.LOCNS
THASP.COUNTER.MAP.SUBSEG.CSV
THASP.COUNTER.MAP.INTERSEC.CSV
THASP.COUNTER.MAP.NODE.CSV
```

This file is created by program THAS754, in traffic volume utility job 760.

5.53 THASP.STATION.VOLUMES

The Station Volumes file contains an extract of the data from the two Traffic Volume files. (See the section on Traffic Volume files above.) It contains the annual and monthly ADTs (Average Daily Traffic) for each Permanent and Short Count station for each desired year. It is created by program THAS755, which is run from the Utilities menu as part of the job that recreates the Segment Volumes file.

```
Sort Order:       Station, Year.
Record Length:    93

Record Definition:

  Columns   Description

   1- 8     Station ID
  10-13     Year
  15-20     Annual ADT
  22-93     12 Monthly ADTs (6 columns each)


The PL/I record definition follows.

  DATASET:   THASP.TEXTLIB
  MEMBER:    THASRSTV

----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+

                        /* Station Volumes file record */
          5 STATION           CHAR (8),
          5 BLANK1            CHAR (1),
          5 YEAR              CHAR (4),
          5 BLANK2            CHAR (1),
          5 ANNUAL_ADT        PIC 'ZZZZZ9',
          5 BLANK3            CHAR (1),
          5 MONTHLY_ADT(12)   PIC 'ZZZZZ9';
```

5.54 THASP.SUBSEG.VOLUMES

This file contains traffic volumes for all sub-segments specified in the THASP.COUNTER.MAP.SUBSEG.CSV file. It is created from the THASP.STATION.VOLUMES and THASP.COUNTER.MAP.SUBSEG.CSV files, by program THAS762, in traffic volume utility job 761.

Each record contains an AADT and 12 MADTs for one year of a sub-segment.

```
Record Length: 117
Sort Order:    Segment, Year, Start_km.

Include file:  THASRSGV:

          /* Segment Volumes file record. Length 117.  THASRSGV */
          5 SEGMENT           CHAR (4),
          5 blank4            CHAR (1),
          5 YEAR              CHAR (4),
          5 blank1            CHAR (1),
          5 START_KM          PIC 'ZZZ9V.9',   /* 6 */
          5 blank2            CHAR (1),
          5 END_KM            PIC 'ZZZ9V.9',   /* 6 */
          5 blank3            CHAR (1),
          5 TYPES             CHAR (2),
          5 ANNUAL_ADT        PIC 'ZZZZZZ9',   /* 7 */
          5 MONTHLY_ADT(12)   PIC 'ZZZZZZ9';   /* 12 * 7 = 84 */
```

Notes:

- A segment may broken up into sub-segments differently in each year.
- For each segment and year, km ranges may not overlap: START_KM must be greater than the previous END_KM.
- The TYPES field shows which types of traffic volume counters were used to derive the data in the record -- Permanent, Short Count, or both ('P ', 'S ', or 'PS'). It is currently not used, but has been stored in case it is ever needed. Data derived wholly or partially from Permanent counters may be somewhat more reliable than data derived from Short Count stations.

5.55 THASP.SUBSET..DATA*

These are accident data subset files created by the H.A.S. Data Retrieval system. Creation, naming and deletion of these files is under user control.

See **Subset Pairs** in the HAS User's Manual / Data Retrieval Concepts.

See **Accident Data Files** at the beginning of the Files section in this manual.

5.56 THASP.SUBSET..DESC*

These are accident data subset description files created by the H.A.S. Data Retrieval system. Creation, naming and deletion of these files is under user control. THASP.SUBSET.name1.DESC describes how file THASP.SUBSET.name1.DATA was created.

See **Subset Pairs** in the HAS User's Manual / Data Retrieval Concepts.

Record length: 80.

5.57 THASP.SWARWTS

This file contains just one record, containing 3 blank-separated integers.
These are the weights to be applied to Fatal, Injury and PDO accidents when calculating Severity-Weighted Accident Rates.

The file can be modified (by privileged users) by editing the file directly, or from an option on the HAS main menu.

The file is read by PL/I routine THASLSW, which stores the weights in external variables FATWT, INJWT and PDOWT defined in TEXTLIB(THASXWTS).

5.58 THASP.TABLE(ACCTYPES)

This file is a list of Counter-Measure Method accident type numbers and descriptions.

This list should be kept consistent with PL/I routine THASSAT, which contains the logic for accident type determination.

```
Record length: 80.
Sort Order:    Accident Type Number

  Columns   Description

   1 -  2   Accident Type Number
   3 -  5   ' - '  (included so that this file can be copied as-is to a
report                          as an accident type key.)
   6 - 80   Accident Type Description

Following is a copy of the file:  (Sept, 1995)

             1 - Right Angle
             2 - Left Turn Opposing
             3 - Straight Ahead Rear End
             4 - Left Turn Rear End
             5 - Right Turn
             6 - Sideswipe
             7 - Head On
```

```
 8 - Off Road
 9 - Fixed Object
10 - Parked
11 - General Rear End
12 - Pedestrian
13 - Animal
```

5.59 THASP.TABLE(ELNETDEF)

"Edit List of NETwork DEFinition files".

This file lists and provides information about all the files which are listed when **Edit Network Definition Files** is selected from the main H.A.S. menu. It is read by REXX program THASEDFL.

There are three records for each file. The first record starts with the file name (without the high level index), and is followed by a description of the file. This information is displayed on panel THASEDFL by REXX program THASEDFL, so that the user can select a file to edit.

The second and third records contain two heading lines which are displayed above the data when the file is being edited on panel THASEDIT.

Note that file THASP.FILEINFO should contain documentation for each of the files listed in TABLE(ELNETDEF).

5.60 THASP.TABLE(ELPARAM)

"Edit List of H.A.S. PARAMeter files"

This file lists and provides information about all the files which are listed when **Edit H.A.S. Parameter Files** is selected from the main H.A.S. menu. It is read by REXX program THASEDFL.

See the description of THASP.TABLE(ELNETDEF) above for details.

5.61 THASP.TABLE(FLDNAMES) - H.A.S. Field Descriptions

The FLDNAMES table describes the fields in the H.A.S. accident records: their locations in the record, and their relationship to fields on the MV104 form.

```
Sort Order: Start byte in HAS record.

  Columns   Description

   1-13     HAS Field Name
            - name of a field in the HAS accident record
            - a PL/I name, as in the record definition

  15-17     Start byte in HAS record

  19-20     Number of bytes in HAS record (field length)

  22-24     MV104 Field Number
            - identifies a field on the MV104 form
            - blank if not from the MV104

  26-26     Field Correspondence - 1: first byte of MV field
                                 - 2: second byte of MV field
                                 - =: same as MV field
                                 - U: unnumbered MV field
                                 - 0: not an MV field

  28-30     Table ID
            - same as the TAB_NUM in file THASP.TABLE(MV104)
            - identifies a set of Data Codes and their Descriptions
            - in many cases, the MV104 Field Number, and the Table ID
              will be the same.
            - where two HAS Fields use the same data codes, those two
              fields will have the same Table IDs.
            - '#  ' if the field is a simple, zero-filled number (e.g.
               TOTALKLD) and does not have codes in THASP.TABLE(MV104).
            - '#b ' same as '#  ' except that the field is blank filled.
            - blank - no codes in THASP.TABLE(MV104).

  32-80     HAS Field Description
            - description of a field in the HAS accident record
            - will usually be the same as an MVB Field description,
              except where an MVB field is split into two HAS fields.


Record definition in THASP.TEXTLIB(THASRFLD):

                                 /* THAS?.TABLE(FLDNAMES) record */
          5 HAS_NAME            CHAR(13),  /* HAS field name */
          5 FILL1               CHAR(1),
          5 START               PIC '999', /* in HAS record */
          5 FILL2               CHAR(1),
          5 LENGTH              PIC '99',  /* in HAS record */
          5 FILL3               CHAR(1),
          5 MV104_NUMBER        CHAR(3),
          5 FILL4               CHAR(1),
          5 CORRESPONDENCE      CHAR(1),   /* between HAS, MVB */
          5 FILL5               CHAR(1),
          5 TABLE_ID            CHAR(3),   /* key to TABLE(MV104) */
          5 FILL6               CHAR(1),
          5 DESCRIPTION         CHAR(49);
```

5.62 THASP.TABLE(MV104) - MV104 Table

TheMV104 Table lists all possible meanings of the codes in the fields of the MV104 form. Records are 80 characters long.

Sort Order: TAB_NUM, TAB_CODE, TAB_EFFECTIVE_DATE.

Record Definition:

Columns	Description		
1- 3	TAB_NUM	-	Field number (e.g. 2A), for fields on the edges of the MV104 form, or DIR for vehicle direction (an unnumbered field).
5- 6	TAB_CODE	-	Code that may occur in this field.
8-15	TAB_EFFECTIVE_DATE	-	Date at which this code (TAB_CODE) in this field (TAB_NUM) began to have this meaning (TAB_DESC).
17-56	TAB_DESC	-	Meaning (description) of this code in this field.

The MV104 Table is read by subroutine THASC01. Subroutine THASTLU searches it for the meaning of a particular code in a particular field. THASTLU is, in turn, called by the Details Report and Summary Report programs in the Data Retrieval subsystem.

The following PL/I record definition is in THASP.TEXTLIB(THASTAB):

```
DCL TAB_RECD                CHAR(80) ;
DCL 1 TAB                   DEF TAB_RECD,
      5 NUM                 CHAR(03),       /* Table Number   */
      5 FILLER1             CHAR(01),
      5 CODE                CHAR(02),       /* Table Code     */
      5 FILLER2             CHAR(01),
      5 EFFECTIVE_DATE      CHAR(08),       /* Effective Date */
      5 FILLER3             CHAR(01),
      5 DESC                CHAR(40),       /* Table Descript */
      5 FILLER4             CHAR(26) ;
```

5.63 THASP.UA.**

These are backup files created by the Update sub-system. (UA comes from Update Archive, but to avoid confusion with the archive master files, they are referred to here as 'backup' files.)

The following are backups of the key files of each update. They are created by update job U40:

```
THASP.UAyyyymm.HSBACC
THASP.UAyyyymm.JUR1.VALLOC
THASP.UAyyyymm.JUR1.INVLOC
THASP.UAyyyymm.JUR2
THASP.UAyyyymm.HSBFATAL
THASP.UAyyyymm.JOBLOG
```

The following are master file backups, and are described in the **Master Files** section of the User's Manual:

```
THASP.UAyyyymm.PROV.VALLOC.CURRENT
THASP.UAyyyymm.PROV.INVLOC.CURRENT
THASP.UAyyyymm.MUN.CURRENT
THASP.UAyyyymm.LOCTEXT.MASTER

THASP.UAJOBU45.PROV.VALLOC.CURRENT
THASP.UAJOBU45.PROV.INVLOC.CURRENT
THASP.UAJOBU45.MUN.CURRENT

THASP.UAyyyy.PROV.VALLOC.ARCHIVE
THASP.UAyyyy.PROV.INVLOC.ARCHIVE
THASP.UAyyyy.MUN.ARCHIVE
```

Prior to 1997, files were of form UAyymmdd.

5.64 THASP.UPGRADE.NOTICE

This is a PDS which contains H.A.S. upgrade notices, one member per notice.

Members are named Nnnn (e.g. N020) where nnn is the sequential upgrade notice number.

See the **Development Environment / Upgrade Notices** section for an example of an update notice.

5.65 THASP.UPyyyymm.HSBFATAL

The Highway Safety Branch Fatal File used to be prepared manually from the fatal MV104 forms by Safety Branch staff. In the **Update Sub-System** section, see **Job U05**, and **Process Fatal Accidents** for information on how the HSBFATAL file is now created.

```
Record Length:    variable.
Sort order:       none: it is sorted before use.

Record Definition:

    Columns Description

     1- 8   Accident Case Number
    10-17   Date (yyyymmdd)
    19-29   Location Code (blank for jurisdiction 2 or 3)
    31      Jurisdiction Code
    32-end  notes
```

The required information takes up the first 31 columns of each record. Subsequent data is ignored by the HAS programs.

It is important that the HSB Fatal file contain records only from the Update period.

The PL/I record is hardcoded in THAS020, as follows:

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+

    /*-------------------------------------------------------------*/
    /* The HSB FATAL record is declared VARYING so that the Highway  */
    /* Safety Branch can change the record length if they wish,      */
    /* without affecting this system  (as long as the first 28       */
    /* bytes remain as defined below).                               */
    /* The FATAL structure is defined on the record at position 3    */
    /* to miss the 2 length bytes at the start of the varying string.*/
    /*-------------------------------------------------------------*/

    DCL HSB_FATAL_RECORD CHAR(200) VARYING;
    DCL 1 FATAL                   DEF HSB_FATAL_RECORD POS(3),
          5 ACCASE      CHAR(8),
          5 BL1         CHAR(1),
          5 ACCDATE     CHAR(8),
          5 BL2         CHAR(1),
          5 LOCN_CODE,
            10 NUMLET   CHAR(3),
            10 SEGNUM   CHAR(4),
            10 KMMARK   PIC '999V9',
          5 BL3         CHAR(1),
          5 JURCODE     CHAR(1);

    DCL FAT_LOCN CHAR(12);
    DCL 1 FAT_LOCN_CODE               DEF FAT_LOCN,
          10 HIGHNUM    CHAR(3),
          10 HIGHLET    CHAR(1),
          10 SEGNUM     CHAR(4),
          10 KMMARK     PIC '999V9';
```

5.66 THASP.UPyyyymm..EDIT Files*

The Edit Files are created by program THAS050.

The fatal edit file THASP.UPyyyymm.FATJUR1.EDIT is created in Update jobs U10 and U15.
The nonfatal edit file THASP.UPyyyymm.NONFAT.EDIT is created in Update jobs U20 and U25.

The files have variable length records. After the fixed portion of the record (defined below), is a variable length string containing the combined Location On and At text fields from the MV104 form.

```
  DATASET:    THASP.TEXTLIB
  MEMBER:     THASREDT

----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+
        2 EDT,
          5 ACTION         CHAR(1),
          5 FILLER1        CHAR(1),
          5 POLICECD       CHAR(4),
          5 POLICE_ERR     CHAR(1),
          5 LOCN_CODE,
            10 HIGHNUM     CHAR(3),
            10 HIGHLET     CHAR(1),
            10 HWY_ERR     CHAR(1),
            10 SEGNUM      CHAR(4),
            10 SEG_ERR     CHAR(1),
            10 KMMARK      PIC'999V9',
            10 KM_ERR      CHAR(1),
          5 LOCN_TYPE      CHAR(2),
          5 TYP_ERR        CHAR(1),
          5 VLINE1         CHAR(1),
          5 PARSE_FLAG     CHAR(1),
          5 VLINE2         CHAR(1),
          5 FILLER2        CHAR(1),
          5 MVB_LOCN       CHAR(11),
          5 FILLER3        CHAR(1),
          5 JURCODE        CHAR(1),
          5 FILLER4        CHAR(1),
          5 ACCASE         CHAR(8),
          5 FILLER5        CHAR(1),
          5 ACCDATE        CHAR(8),
          5 FILLER6        CHAR(1),
          5 PLACE          CHAR(10), /* added 2001-05-30 */
          5 FILLER7        CHAR(1),
          5 VEHDIR1        CHAR(1),  /* added 2001-10-29 */
          5 FILLER8        CHAR(1);  /* Location text: (ON,AT) not included


Note: a variable length string containing Location On and At text follows
FILLER8.
```

5.67 THASP.UPyyyymm.LOCTEXT

This file contains the PLACE, ON and AT text information, which is provided on the MVB accident record, but is not stored in the HAS accident record.

The file is created from the THASP.UPyyyymm.MVB file by program THAS012, in job U01, at the same time as the HSBACC file is created. It is used later in the update process, by program THAS085 in job U38

Sort Order: Case

Record defined in THASP.TEXTLIB(THASRLTX), as follows:

```
 /* Record definition for Location Text (Place-On-At) data */

    DCL LOCTEXT_REC CHAR(126) STATIC;
    DCL LOCTEXT_KEY CHAR(16) DEFINED(LOCTEXT_REC);

    DCL 1 LTX DEFINED(LOCTEXT_REC),
          5 ACCASE  CHAR(8),
          5 ACCDATE CHAR(8),
          5 PLACE   CHAR(10),
          5 ON      CHAR(50),
          5 AT      CHAR(50);
```

5.68 THASP.UPyyyymm.LOCTEXT.VALLOC

This file is a copy of THASP.UPyyyymm.LOCTEXT reduced to those accidents on the JUR1.VALLOC file. Produced by program THAS085 in job U38.

Sort order: Case

5.69 THASP.UPyyyymm.MVB

For each update, this is the file of accident data obtained from the Motor Vehicle Branch. It is read by update job U01.

The record layout of the data obtained from the Motor Vehicle Branch was revised in May 2001. At this time the MVB changed to use a query on their Oracle database, instead of a query which simulated their previous database.

The Pre-April 2001 record layout, used by program THAS010 is in THASP.TEXTLIB(THASRMVB)
The Pre-April 2004 record layout, used by program THAS011 is in THASP.TEXTLIB(THASRMV2)
The current record layout, used by program THAS012, is in THASP.TEXTLIB(THASRMV3) (listed below).

Used till	TEXTLIB	Length	used by program	Notes
April 2001	THASRMVB	640	THAS010	
April 2004	THASRMV2	636	THAS011	Location text added
	THASRMV3	679	THAS012	PLFN & other new fields

Among other changes, the new format includes textual location information.

```
Sort Order:    Incident (Case) Number
```

See the Update section for information on how the MVB file is obtained.

```
  DATASET:    THASP.TEXTLIB
  MEMBER:     THASRMV3
```

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+

          3 PAGE_NO        CHAR(1),  /*PAGE NO WITHIN ACC           */
          3 INO            CHAR(8),  /* INCIDENT NUMBER             */
          3 ADT            CHAR(6),  /* ADDED DATE                  */
          3 IDT            CHAR(6),  /*INC DATE YYMMDD              */
          3 ITM            CHAR(4),  /*INCIDENT TIME                */
          3 ITY            CHAR(1),  /*INC TYPE                     */
                                     /*1=NON REPORT, 2=PROP DAMAGE  */
                                     /*3=INJURY, 5=FATAL            */
          3 HR             CHAR(1),  /*6=YES , 0=NO                 */
          3 AT             CHAR(1),  /*1=ATTENDED,2=NOT ATTENDED    */
          3 PCD            CHAR(4),  /* POLICE CODE                 */
          3 PZN            CHAR(3),  /*POLICE ZONE                  */
          3 PLFN           CHAR(16), /*POLICE FILE NUMBER           *2004*/
          3 LCD            CHAR(12), /*LOCATION (DISTRICT) CODE     */
          3 PLC            CHAR(10), /*PLACE NAME                   */
          3 DG             CHAR(2),  /*DIAGRAM CODE NO              */
          3 TIN            CHAR(3),  /*TOTAL INJURED                */
          3 TKL            CHAR(3),  /*TOTAL KILLED                 */
          3 TVH            CHAR(3),  /*TOTAL VEHICLES               */
          3 RCL            CHAR(2),  /*ROAD CLASS (COND 1)          */
          3 TFL            CHAR(2),  /*TRAFFIC FLOW  (1A)           */
          3 ILC            CHAR(2),  /*ACC LOCATION (COND 2)        */
          3 SZN            CHAR(3),  /*SPEED ZONE    (2A)           */
          3 SZNADV         CHAR(3),  /*Advisory SPEED ZONE (2B)     *2004*/
          3 LUS            CHAR(2),  /*LAND USAGE (COND 3)          */
          3 RSF            CHAR(2),  /*ROAD TYPE (COND 4)           */
          3 TCT            CHAR(2),  /*TRAFFIC CONTROL (COND 5)     */
          3 RCH            CHAR(2),  /*ROAD CHARACTER (COND 6)      */
          3 RCO            CHAR(2),  /*ROAD SURF COND (COND 7)      */
          3 WEA            CHAR(2),  /*WEATHER COND (COND 8)        */
          3 LGH            CHAR(2),  /*LIGHTING COND (COND 9)       */
          3 T2C            CHAR(2),  /*TYPE COLLISION 2ND EVENT (21)*/
          3 LEV            CHAR(2),  /*LOC 1ST EVENT COND 24        */
          3 PLOC           CHAR(2),  /*PED LOC COND 29              */
          3 PEAC           CHAR(2),  /*PEDESTRIAN ACTION COND30     */
          3 SS             CHAR(2),  /*SPECIAL STUDY COND 37        */
        3 ENTITY(2),
             5 CAT         CHAR(1),  /*V=VEHICLE,P=PEDESTRIAN,      */
                                     /*C=CYCLIST, N=NONE            */
             5 ENO         CHAR(2),  /*ENTITY NUMBER                */
             5 POF         CHAR(1),  /*POSSIBLE CAUSOR 1=YES,2=NO   */
             5 CF1         CHAR(2),  /*CONTRIBUTING FACTORS         */
             5 CF2         CHAR(2),     /*1-COND 31,32,33,33A       */
             5 CF3         CHAR(2),     /*2-COND 34,35,36,36A       */
             5 CF4         CHAR(2),  /*                             *2004*/
        3 DR_#        PIC'9',        /* # driver recs filled        */
        3 DRIVER(2),
             5 DPRO        CHAR(2),  /*DRIVER LIC PROV/STATE        */
             5 LCL         CHAR(3),  /*DRIVER CLASS                 */
             5 NL          CHAR(1),  /*NOVICE/LEARNER- N,L or blank *2004*/
             5 BDT         CHAR(6),  /*DRIVER BIRTHDATE  YYMMDD     */
             5 ESEX        CHAR(1),  /*DRIVER SEX                   */
             5 BRE         CHAR(2),  /*DRIVER BREATH TEST           */
             5 CH1         CHAR(9),
             5 CH2         CHAR(9),  /*TYPE OF CHARGE               */
             5 CH3         CHAR(9),  /*SECTION OF ACT,BYLAW,ETC     */
        3 VEH_#       PIC'9',        /* # vehicle recs filled       */
        3 VEHICLE(2),
             5 VPRO        CHAR(2),  /*PLATE PROV-STATE             */
             5 NSC         CHAR(16), /*NATIONAL SAFETY CODE NUMBER  */
```

```
    5 DIR        CHAR(1),  /*DIRECTION OF TRAVEL          */
    5 DLO        CHAR(2),  /*VEH DAMAGE LOCATION         */
    5 DSE        CHAR(2),  /*VEH DAMAGE SEVERITY         */
    5 T3C        CHAR(2),  /*TYPE COLLISION 3RD EVENT    */
                           /* COND 22,23                 */
    5 PRAC       CHAR(2),  /*VEH PRE ACC ACTION COND 25,26*/
    5 VTY        CHAR(2),  /*VEH TYPE COND 27,28         */
    5 VUS        CHAR(2),  /*VEHICLE USAGE COND 27A,28A  */
    5 STOLEN     CHAR(01), /* Y or N                     *2004*/
    5 VYR        CHAR(2),  /*VEHICLE YEAR               */
    5 VMK        CHAR(10), /*VEHICLE MAKE               */
    5 VST        CHAR(5),  /*VEHICLE STYLE              */
    5 VIC        CHAR(6),  /*VEHICLE ID CODE            */
    5 VREG       CHAR(7),  /*VEHICLE REGISTRATION NUMBER */
3 VICT_#      PIC'9',      /* # victim recs filled       */
3 VICTIM(8),
    5 VOC        CHAR(2),  /*WHICH VEH OCCUPIED  (10)   */
    5 POS        CHAR(2),  /*POSITION IN/ON VEH  (11)   */
    5 SAF        CHAR(2),  /*SAFETY EQUIP USED  (12)    */
    5 EJE        CHAR(2),  /*EJECTION FROM VEH  (13)    */
    5 AGE        CHAR(3),  /*AGE OF VICTIM  (14)        */
    5 VSEX       CHAR(1),  /*SEX OF VICTIM  (15)        */
    5 LCI        CHAR(2),  /*LOC OF INJURIES  (16)      */
    5 TYI        CHAR(2),  /*TYPE OF INJURIES  (17)     */
    5 INJCLASS   CHAR(2),  /*INJURY CLASS               *2004*/
    5 STA        CHAR(2),  /*VICTIM STATUS  (18)        */
    5 TTO        CHAR(4),  /*HOSP ETC. TAKEN TO  (19)   */
    5 TBY        CHAR(4),  /*METHOD OF TRANS TO HOSP ETC */
                           /*               ( COND 20)  */
3 LOC_ON         CHAR(50), /* Location ON text          */
3 LOC_AT         CHAR(50); /* Location AT text          */
```

5.70 THASP.UPyyyymm.JOBLOG

For each update, a log of all the Update jobs run is automatically maintained in a file. The file is called THASP.UPyyyymm.JOBLOG (where yyyymm is the date of the update). See the User's Manual for an example job log file.

The file has record length = 80.

The first record of the file is a title record, containing the date of the Update. This title is inserted when the file is created by REXX program THASUPD.

There are normally two log records for each job run. The first one is the job submission record. It is put on the file by the REXX program THASUSEL. It contains the following fields:

```
01-14 -  Date and time of the job submission
16-18 -  'JOB'
20-22 -  the three character job name (eg 'U80')
24-25 -  the submission number of this job: '04' if this is the fourth
         submission of this job in this update.
37-34 -  the job name, as appears on the listing banner page.  For job U80
         this will be 'THASPU80', unless USERID job names have been
         requested.
36-40 -  the job number, as appears on the listing banner page.
42-48 -  the user ID under which the job was submitted.
50-80 -  'SUBMITTED'
```

REXX program THASUPD puts the JCL from JCLIB member THASJLOG onto the end of the JCL of each Update job it submits. Depending upon how the job went, a step will execute program THAS071, which will write a Job Log record with a given completion message to the Job Log file (and to the job listing).

The job completion record is the same as the submission record with the following exceptions:

```
    - the date and time are the date and time the job completed
    - the job number and userid are omitted
    - instead of 'SUBMITTED', the record will end with:
         'COMPLETED SUCCESSFULLY'   -    if all went well
         'ERROR: COND > 8'          -    if any step in the job had a
                                         completion code greater than 8
         'TERMINATED ABNORMALLY'    -    if an error occurred which caused a
                                         system ABEND.
```

If there was a JCL error in the job, no completion record is produced - on the listing or in the job log.

5.71 <userid>.(VOLUME).CSV

These are temporary files, created by program THAS752 for downloading to a PC. After downloading, the file should be deleted from the mainframe.

"VOLUME" is the default name used. The user may specify another name.

```
Sort order: Segment, Start_km.

Record layout: (hard-coded in THAS752)

    DCL OUTREC              CHAR(34);
    DCL 1 OUT DEF OUTREC,                   /* length 34 */
          5 SEGMENT     CHAR(4),
          5 comma1      CHAR(1),
          5 START_KM    PIC'ZZZ9V.9',
          5 comma2      CHAR(1),
          5 END_KM      PIC'ZZZ9V.9',
          5 comma3      CHAR(1),
          5 ADT         PIC'ZZZZZ9',
          5 comma4      CHAR(1),
          5 NODENAME    CHAR(8);
```

6 REXX/ISPF Dialog

6.1 Introduction

The user selects, creates and submits Highway Accident System batch jobs using an ISPF Dialog. The ISPF Dialog consists of REXX programs, ISPF Panel definitions and JCL Skeleton files.

The REXX programs control the sequence of events, define variables, and invoke the ISPF services: Panels, Variable services, Table services and File Tailoring services. All ISPF service calls in the REXX code are preceded by "ISPEXEC".

ISPF Panels contain the menu and data entry screen layouts, and some data verification logic. A Panel may also obtain a message from a message file to display on the screen.

Variables reside in variable Pools. Function Pool variables are available to a REXX subroutine and the panels it invokes. Shared Pool variables are global: available in any panel, and in any REXX program via a VGET request. Profile Pool variables are like Shared Pool variables, except that they are kept between sessions (stored on disk).

Tables are used to store multi-line tables of information. Tables can be displayed in scrollable areas in Panels, using the ISPF TBDISPL service. Tables can also be stored on disk, but the Highway Accident System uses mostly temporary tables.

Skeleton Files contain JCL with embedded variable names. The ISPF File Tailoring service is used to read a JCL skeleton file, substitute values for variable names, and write (or append) the result to a result file. The result file of JCL can then be submitted as a batch job.

6.2 ISPF Libraries

The ISPF source files are stored in libraries named `hli.ISPF.ISPxLIB,` where 'x' may be:

```
C - REXX programs
P - Panels
S - Skeletons
M - Messages
T - Tables
```

and 'hli' may be THASD for development and THASP for production.

All member names start with "THAS" to avoid member name collisions with members of other concatenated libraries at execution time. Panel members are generally named the same as the REXX programs which invoke them. The exceptions (of which there are few) occur when a REXX program uses more than one panel.

At design time there were ministry-wide ISPF libraries available, but dedicated Highway Accident System libraries were used for improved security, improved productivity, and simpler member promotion procedures.

6.3 User Setup and RACF for Starting

6.3.1 Overview

Setup for new HAS users is done by the Information Systems Branch. (Dean Rogers).

Users should be in RACF group THASP (headquarters) or THASP1 (others).

TSO :

```
PROCEDURE ==> SPFAUTO
COMMAND   ==>                              (blank)
```

This causes the following chain of procedures to execute:

SYS9.SYSPROC(SPFAUTO)	
<userid>.CLIST(INIT)	- a copy of THASP.REXX.EXEC(USERINIT) - customizable by the user
THASP.REXX.EXEC(SETUPDF)	- allocates SYSEXEC to THASP.REXX.EXEC
THASP.REXX.EXEC(HASPDF)	- starts PDF with HAS libraries allocated.

TSO Logon Alternative:

```
PROCEDURE ==> SPF
COMMAND   ==> THASP.REXX.EXEC(SETUPDF)
```

This is more direct, but does not allow for customization in <userid>.CLIST(INIT)

THASP.REXX.EXEC(USERINIT):

```
/* REXX ***************************************************************
|
|  CLIST(INIT) for Highway Accident System (HAS) users.
|
|  - starts PDF with THASP libraries allocated.
|
|  Coding SPFAUTO in the PROCEDURE field of the TSO logon screen
|  causes SYS9.SYSPROC(SPFAUTO) to execute, which then executes
|  <userid>.CLIST(INIT).
|
|--------------------------------------------------------------------
|  1997 Apr 11  Matthew Nicoll, Cypher Consulting, for BC MoTH
|********************************************************************/
"EXEC 'THASP.REXX.EXEC(SETUPDF)'"
```

SETUPDF is described in a following section.

6.3.2 ISB Setup Procedure for New HAS User

(This information received from Dean Rogers in June 2004)

Security receives a Change Management Request from HQ or a Region send to approver Jerry Froese.

A. User does not currently have an SCxxxxx id.

Since the ORS system has been discontinued, it has become necessary to fax a completed MVS Userid Authorization form to MVS Admin for all new userids and for re-activations of existing inactive userids. This form must be signed by the user and their Manager, can be faxed back to us, and must then be signed by the GDSA and faxed to MVS Admin at 250-387-9651.

RACF default group = THASP, Authority(C) for HQ or THASP1, Authority (C) for Regions

Default charge no. = 606096
Item Notes: TSO required <enter>
Charge Number = 151271 for Disk Space (DASD) and TSO <enter>

Follow Status of on line request notes in GDSA Function

When id. received, use the GDSA id. to unrevoke it and set the password.
Type: tso gd69547 <Enter>
Type: password <Enter>
*** <ENTER>

Type: **6 <Enter>**
Type: alu sc###### password(_____) resume

Granting permissions to the new HAS userid.

Then, permit the group THASPZ to the user's generic profile, with ALTER access, by keying in, from native TSO:
<type>: PE 'Scxxxx.**' ID(GD69547) ACCE(ALTER) <enter> - to give Security access to the id's profile, and,
<type>: PE 'Scxxxx.**' ID(THASPZ) ACCE(ALTER) <enter> - to give the audited mtce. group THASPZ access.

Hit **F3**
From the ITSD ISPFPrimary Option Menu **Type: 3.3 <enter>**
Hit **F8** and in the Move/Copy Utility Screen under Option: Type **C** (for Move/Copy)
TAB to *Other Sequential or Partitioned Datatbase.*
<type>: **'THASP.REXX.EXEC(USERINIT)'** <enter>
Tab to other Partioned or Dataset
<type>: **'sc*****.clist(init)'** <enter>
[Message: Confirmation on upper right of screen: Member userinit replaced]
(This sets up the HAS environment automatically for the user when they sign on.)
Log off as the GDSA.
To log-off Hit **F3**
<type>: X <enter>
<type>: Z <enter>

At the Primary Screen log-on as the user and remove the function key displays. This cannot be done from the GDSA id., so it is necessary **to sign on as the user**.
<type:> **TSO SC ******* <enter>
<type:> **start-up password** ☺ tab
to NewPassword and <type>: password **M*****
tab to Procedure: <type> **SPFAUTO**
tab 7 times to No-Notice and <type> **S**,
system automatically moves you to Reconnect, <type: **S**> <enter>
You will prompted to verify password.
<type>: **M******** (new password) for verification:<enter>
when you see ******* <enter>
You are now back at ISPF Primary Option Menu signed on as the user.

To remove ISPF Commands go to Options, left mouse click to hilite and <enter> to open
<type> **1** – General settings in that field, <enter>
left mouseclick on Function Keys to hilite, <enter>
<type> **6** - to Remove Function Key Display, <enter> Function Keys are removed.
Hit **F3** to return to Primary Option Menu.
<type>: **IOFR** <enter> to reach the IOF Job Access screen.
In Option: <type>: **G** Tab thru fields to Job Name or Pattern: <type>: *
Tab to Userid or Group which Access modified and <type>: **THASPZ**
Tab to Access to be Granted and <type>: **browse** <enter>
Look to the upper right-hand of screen for **GRANT SUCCESSFUL**.
Hit **F3** to return to Primary Option menu.

You must return the function keys. Simply repeat the "remove function keys" steps, but after left mouse click on Function Keys to hilite, <enter>
<type>: **4 Show All Function Keys** <enter>
Hit **F3** to return to Primary Option Screen. Function Keys should return to bottom of screen. Logoff: <type>: **X** <enter>
<type>: **Z** <enter> You are returned to Primary Screen.

Logon with GDSA id. to reset the password for the new id.

Go to D

B. User has an SCxxxx id., but does not have TSO access
To add TSO access to an existing userid, fax a completed MVS Userid Authorization form to MVS Admin with "Add TSO access" in the **Notes** section of the form. This form must be signed by the user and their Manager, can be faxed back to us, and must then be signed by the GDSA and faxed to MVS Admin at 250-387-9651.
Charge Number = 151271 for Disk Space (DASD) and TSO <enter>

Follow Status of on line request notes in GDSA Function

When id. received, use the GDSA id. to connect the group
Group = **THASP**, Authority(Create) for HQ or **THASP1**, Authority (Create) for Regions
Note that this must be set as the user's default RACF group, using the following command:

alu sc##### dfltgrp(THASP)
or

alu sc##### dfltgrp(THASP1)

Type: tso gd69547 <Enter>
Type: password <Enter>
*** <ENTER>
Type: **6 <Enter>**

Type: co sc###### group(_____) auth(create)

Following the instructions: Granting permissions to the new HAS userid.

Call user that you will sign-on with their userid and that you will assign a new password for them to sign-on with when HAS access is complete.

Go to D.

C. User has an MVS id. with TSO access.
Security connects the id. to the group THASP(C) if a HQ user, THASP1(C) if a Regional user. Depending on user's other groups, and current use, decide on which should be the default group and let user know. See diagram for TSO logon parameters NONOTICE and RECONNECT. Update HAS user list.

D. When complete forward UserId to reporter name and other interested parties
cc Jerry Froes and Matthew Nicoll.
Email to include: Name, UserId. User can expect some introductory information from Matthew Nicoll. <send email>
Email user with their userid and their initial password.

In remedy go to the Supporting Information Tab and select Security Tab; Mainframe Sub Tab, fill in Mainframe Id and update work log and close request. File request in mainframe folder when complete.

Add new userid and name to HAS UserId list manually. When many changes are made update list and email new list to HAS contacts, Brad, Sam, Matthew and Jerry.

TASKS of Application support contact: Matthew E Nicoll

*(email: **menicoll@CypherConsulting.com** [XT;NICOLL, MATTHEW])*
- add the new user to my HAS users mailing list,
- add the new UserId to a UserId list and upload it to the mainframe (so that I can find out whether any HAS users are logged on)
- verify that the UserId.CLIST(INIT) file was set up properly.
- send the new user some introductory information.

```
------------------------------ TSO/E LOGON ----------------------------


    Enter LOGON parameters below:                RACF LOGON parameters:

    UserId    ===>
    Password  ===>                                 New Password ===>

    Procedure ===> SPFAUTO                         Group Ident  ===>

    Acct Nmbr ===> 151271

    Size      ===> 2048

    Perform   ===>

    Command   ===>

    Enter an 'S' before each option desired below:
            -Nomail       S -Nonotice      S -Reconnect        -OIDcard

PF1/PF13 ==> Help  PF3/PF15 ==> Logoff PA1 ==> Attention  PA2 => Reshow
You may request specific help information by entering a '?' in any entry field
       ------------------------------------------------------------------
```

6.4 Startup REXX Programs

The following REXX programs are stored in PDS THASP.REXX.EXEC, and are used to setup and start the H.A.S. ISPF Dialog:

SETUP (normal for developers)

```
/* SETUP  ************************************************** REXX
|
|  This REXX program allocates PDS THASD.REXX.EXEC to DDNAME
|  SYSEXEC, and tells TSO to look in SYSEXEC for CLIST & REXX programs.
|
|  Once this program has been executed, the Highway Accident System
|  can be started from native TSO with command %HAS, and from
|  within PDF with command TSO %HAS.
|
|-------------------------------------------------------------------
| 91/06/24 | Matthew Nicoll, Cypher Consulting
*******************************************************************/

"EXECUTIL SEARCHDD(YES)"          /* Tells TSO to look in SYSEXEC    */
                                  /* for REXX execs named "%NAME"    */

"ALLOCATE DDN(SYSEXEC) SHR REUSE DSN('THASP.REXX.EXEC')"
```

SETUPDF - SETUP + HASPDF (normal for users)

```
/* SETUPDF ************************************************** REXX
|
|  This REXX program allocates PDS THASD.REXX.EXEC to DDNAME
|  SYSEXEC, and tells TSO to look in SYSEXEC for CLIST & REXX programs.
|
|  It then starts PDF.
|
|  Once this program has been executed, the Highway Accident System
|  can be started from native TSO with command %HAS, and from
|  within PDF with command TSO %HAS.
|
|-------------------------------------------------------------------
| 91/07/03 | Matthew Nicoll, Cypher Consulting
*******************************************************************/

"EXECUTIL SEARCHDD(YES)"          /* Tells TSO to look in SYSEXEC    */
                                  /* for REXX execs named "%NAME"    */

"ALLOCATE DDN(SYSEXEC) SHR REUSE DSN('THASP.REXX.EXEC')"

"%HASPDF"
```

HASPDF

- starts PDF with the H.A.S. ISPF libraries concatenated to the system ISPF libraries, so that the H.A.S. ISPF Dialog can be started from within PDF.
- to get Development (THASD) libraries for the REXX programs and menus, a 'D' must be specified as a parameter. That is, use **%HASPDF D** instead of **%HASPDF**.

HAS

- starts the H.A.S. ISPF Dialog, from native TSO, or from within PDF.
- PDF must have been started with HASPDF for this to work from within PDF.

6.5 REXX Program Descriptions

The following REXX programs are in THASP.ISPF.ISPCLIB.

Name (Calls)	Description
THAS720	- Displays a panel to allow the user to specify modifications to the master file.
THASTFLD	
THAST720	
THASALLS	- Allocate a small data set
THASASUB	- Subset data source for data retrieval step A
THASSSEL	
THASATAB	- Creates table ARFDSN (Average Accident Rate files)
THASCHAT	- Check counter-measure accident types, for 210 & 220
THASCLAS	- Select accidents by highway class
THASNTAB	
THASVALC	
THASCMPR	- Compress a PDS
THASCOMB	- Combined Hazardous Locations/Sections Report (OBSOLETE, not used)
THASCONF	- Confirmation to submit a job
THASDIST	- Select accidents by highway district
THASNFIL	
THASDRET	- Data Retrieval main program
THASKOPN	
THASP200	
THASASUB	
THASPSEL	
THASIJCD	
THASCONF	
THASDSEL	- Data source selection
THASDSOK	- checks to see if a dataset exists, and whether it is migrated (rolled out)
THASEARF	- Display the Average Rate File table ARFTAB for editing.
THASEDFL	- Select an H.A.S. file to edit
THASFRTO	- Select accidents by From-To specifications
THASFSEL	- Select accidents by data field
THASTFLD	
THASSMVB	
THASDIST	
THASCLAS	

```
THASHLI             - Displays the high level index panel THASHLI.

THASIJCD            - Insert job card JCL on front of file tailoring output.
    THASJNAM
    THASPRTY

THASJCLS            - Append JCL for one Data Retrieval process
    THASSUBJ

THASJNAM            - Creates job and skeleton names

THASKOPN            - Prepare for and start the JCL Skeleton file tailoring
    THASALLS

THASLOGI            - Checks the critical criteria logic (obsolete: NOT USED)

THASMAIN            - H.A.S. Main program
    THASDRET
    THASEDFL
    THASHLI
    THASPARM
    THASSSCL
    THASSWTS
    THASUPD
    THASUUTL
    THASUTIL

    THASEDFL

THASMSEL            - Member selection

THASNFIL            - Right-justify a number with specified fill character

THASNTAB            - Create Class Names table CLASSNAM from dataset
                      THASP.CLASS.NAMES

THASP200            - Data Selection specification
    THASALLS
    THASMSEL
    THASCLAS
    THASDIST
    THASPECS
    THASJCLS
    THASNFIL
    THASCMPR
    THASFRTO
    THASFSEL

THASP203            - gets a data field selection spec (query)
    THASALLS
    THASPECS
    THASJCLS
    THASFSEL
    THASMSEL
    THASCMPR

THASP205            - creates a step for running program THAS205
                      (select by counter-measure accident type).
    THASPCMT
    THASPECS
    THASJCLS

THASP210            - Create step for Accident-prone Locations
```

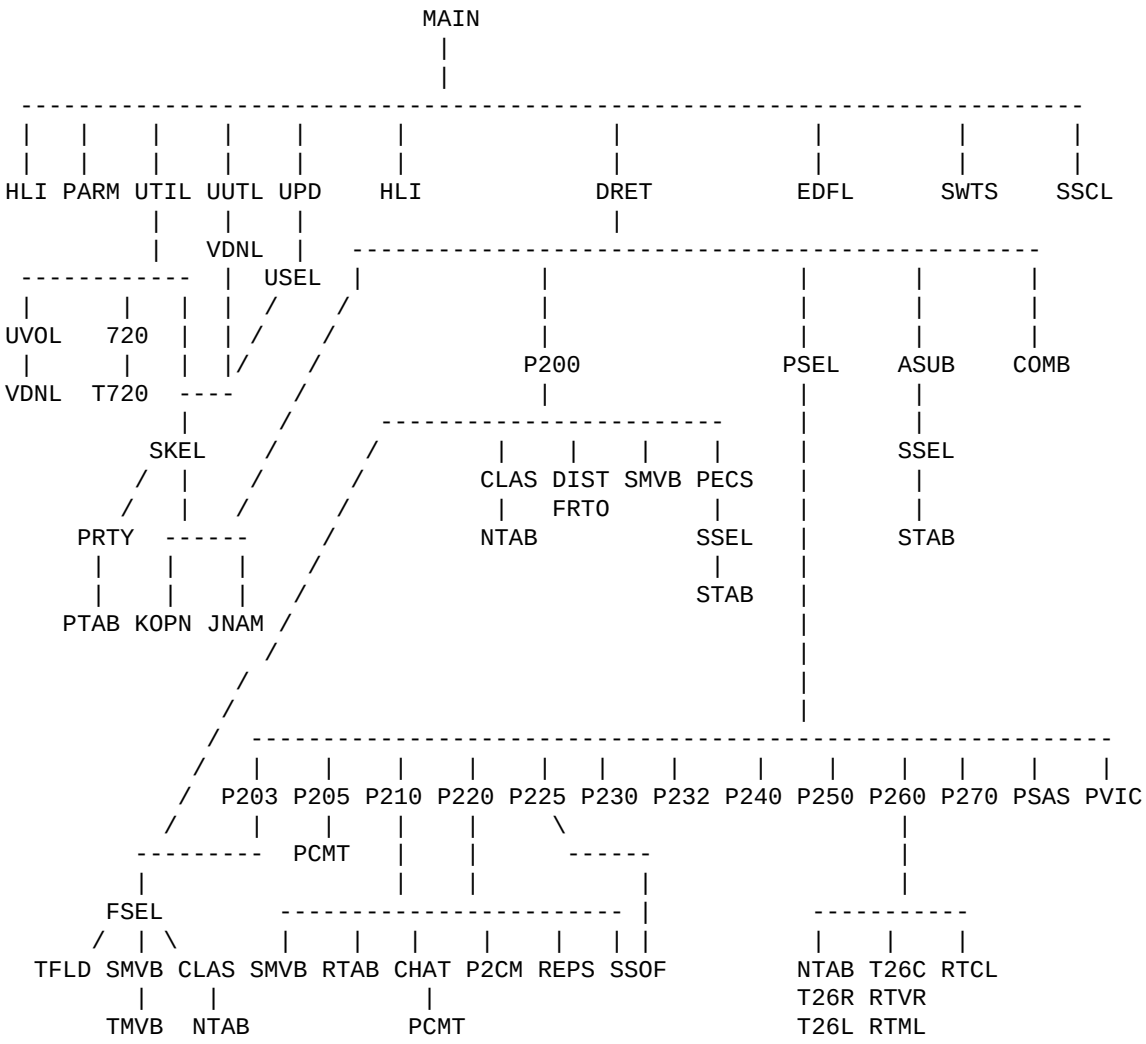
```
    THASRTAB
    THASREPS
    THASPECS
    THASP2CM
    THASCHAT
    THASMSEL
    THASSMVB
    THASSATR
    THASPARF
    THASSSOF
    THASJCLS

THASP220            - Create step for Accident-Prone Sections
    THASRTAB
    THASREPS
    THASPECS
    THASP2CM
    THASSATR
    THASCHAT
    THASMSEL
    THASPARF
    THASSSOF
    THASSMVB
    THASJCLS

THASP225            - Create step for Specified Section Analysis
    THASSSOF
    THASPARF
    THASPECS
    THASDSOK
    THASMSEL
    THASJCLS

THASP230            - Create step for the Histogram Report
    THASJCLS

THASP232            - Create step for Accident Counts
    THASJCLS

THASP240            - Create step for the Details Report
    THASJCLS

THASP250            - Create step for the Summary Report
    THASJCLS

THASP260            - Rate Table
    THASNTAB
    THAST26C
    THAST26R
    THAST26L
    THASRTCL
    THASRTVR
    THASRTML
    THASJCLS

THASP270            - Average Accident Type Ratios
    THASPECS
    THASJCLS

THASP2CM            - Display counter-measure method panel for 210 & 220
    THASPCMT

THASPARF            - Main routine for Selecting and/or Editing an
```

```
                    Average Accident Rate File.
    THASSARF
    THASTARF
    THASEARF

THASPARM            - Profile pool parameters - display & set

THASPCMT            - Make sure that the Counter-Measure accident type
                      descriptions  are defined in variables
                      AT1DESC - AT13DESC in the shared pool.

THASPECS            - Output subset specification
    THASSSEL

THASPRTY            - Job priority & time
    THASPTAB

THASPSAS            - SAS program specification
    THASMSEL
    THASPECS
    THASJCLS

THASPSEL            - Data retrieval process selection
    THASP210
    THASP220
    THASP230
    THASP240
    THASP250
    THASP260
    THASP270
    THASP232
    THASP205
    THASP203
    THASP225
    THASPSAS
    THASPVIC

THASPTAB            - Create the job priority & time table

THASPVIC            - Create a Victim file, and run a SAS program on it.

THASREPS            - Report sort specification

THASRTAB            - Create the report sort table

THASRTCL            - Define Rate Table columns (highway classification sets)
    THASVALC

THASRTML            - Specify Rate Table maximum section lengths

THASRTVR            - Define Rate Table rows (volume ranges)

THASSARF            - Display a list of Average Rate Files

THASSATR            - Display a list of Accident Type Ratio file pair names

THASSSCL            - Display for selection a list of highway classification
                      scheme names

THASSKEL            - Start JCL skeleton processing for a job
    THASJNAM
    THASKOPN
    THASPRTY
```

```
THASSMVB            - Select MV104-form codes for one field
    THASTMVB

THASSSEL            - Subset selection
    THASSTAB

THASSSOF            - display and allowing user-selection and ordering
                      of Specified Section Analysis (THAS225) Output Fields.

THASSTAB            - Create the subset table

THASSUBJ            - Define JCL parameters for one subset

THASSWTS            - Display for editing the weights from file THASP.SWARWTS

THAST26C            - Open or create Column Definitions table

THAST26L            - Open or create Section Lengths table

THAST26R            - Open or create Row Definitions table

THAST720            - Opens or creates table THAS720, of master file
                      modification specifications.

THASTARF            - Creates table ARFTAB with the contents of a specified
                      Average Rates File.
THASTFLD            - Create field name table from THASP.TABLE(FLDNAMES)
THASTLMK            - Creates a table of landmark codes and descriptions from
                      file THASP.LANDMARK.TYPES
THASTMVB            - Create code table for one field from file
                      THASP.TABLE(MV104)
THASTPLC            - Creates a table of police detachement codes and
                      descriptions, from file THASP.RCMP.DETACH
THASTREG            - Creates a table of region codes and descriptions.
THASTTAB            - Creates table ATRDSN  with the names of all Accident
                      Type Ratio file pairs for the current user.

THASUPD             - Update main program
    THASALLS
    THASUSEL

THASUSEL            - Update job selection
    THASSKEL
    THASCONF

THASUTIL            - Utilities main program
    THASPARF
    THASUVOL
    THASSKEL
    THASCONF
    THAS720

THASUUTL
    THASVDNL
    THASPARF

THASUVOL            - Traffic Volume Utilities
    THASVDNL
    THASSKEL
    THASCONF

THASVALC            - Validate highway classification codes
```

```
THASVCLS            - Verifies that a Highway Classification Set is valid.

THASVDNL            - Create a traffic volume file for downloading to a PC
    THASSKEL
    THASCONF
```

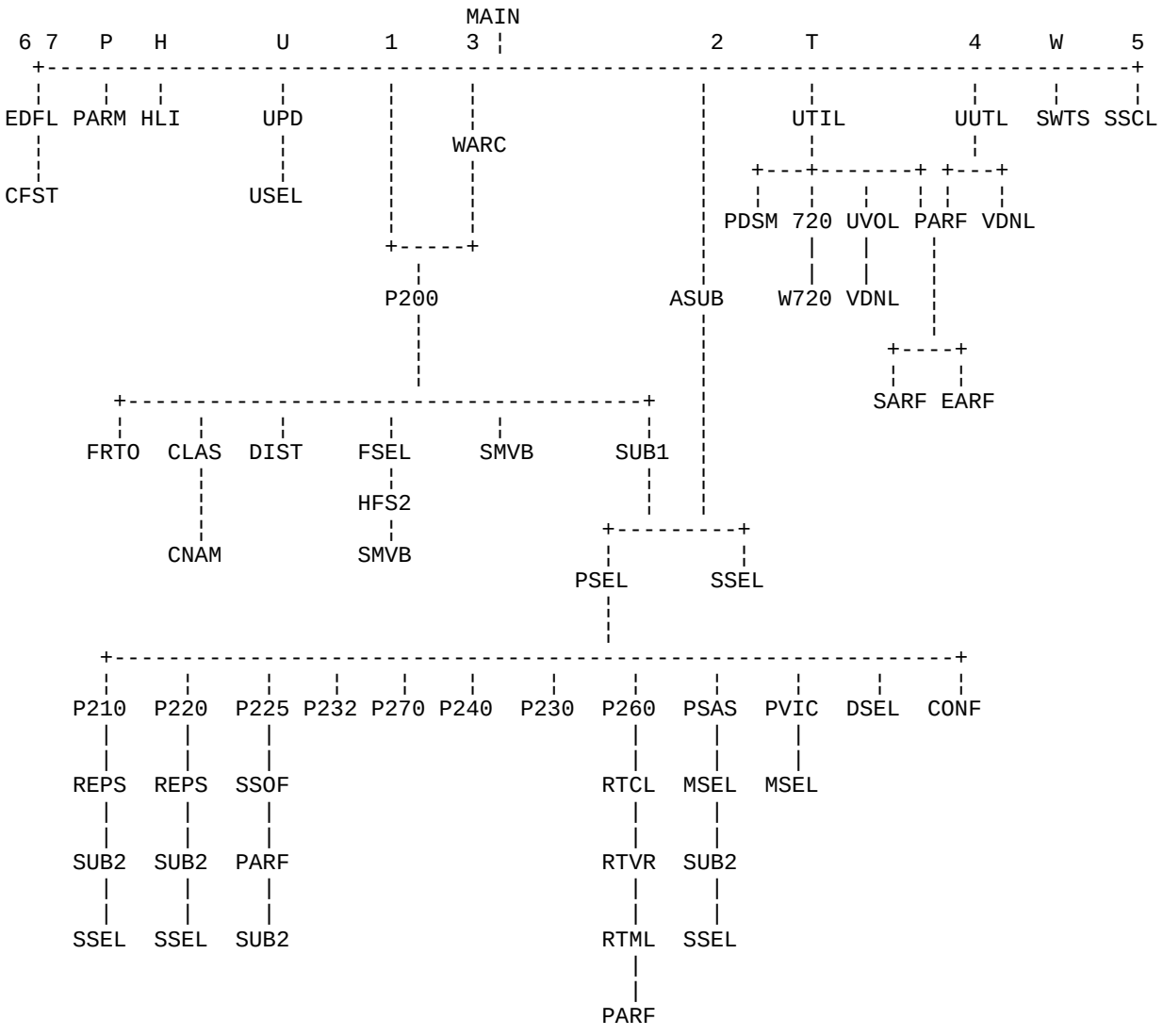
6.6 REXX Structure Chart



Utility routines which are called by many other routines are not included above.

6.7 Panel Organization Chart

This diagram shows the order in which menus and panels may appear.
Help panels are not shown.
Broken vertical lines show panels that are called in order from a single panel; solid lines show panels that may be called in any order.



6.8 High Level Indexes

All dataset high level indexes are coded as variables in the JCL skeleton files. The variables are:

```
DSHLI       - H.A.S. Data Sets, including all created (new) datasets
PROCHLI     - PROCLIB (Cataloged Procedure library)
LOADHLI     - LOADLIB (Executable PL/I program library)
CARDHLI     - CARDLIB (library of Sort control statements etc.)
LKIHLI      - all read-only LKI and LKI-related datasets
PDSMHLI     - PDS-Master files: .SEG.ACDAT, .NODE.ACDAT, .PDSM.DATELIMS
              - note: does NOT include .SHNFIL
```

These variables are kept in each user's Profile variable pool. They can be accessed by selecting H on the H.A.S. Main Menu. A developer can set the high level indexes to select the files desired for development and testing purposes. Users should ordinarily not change any high-level indexes.

See REXX program THASHLI, and panel THASHLI.

The DSHLI variable is also used in the REXX programs, for files such as <dshli>.DATASEL.

7 Update Sub-System

7.1 Introduction

The purpose of the Highway Accident System Update Sub-System, is to update the Highway Accident System master files with accident data from the ICBC Traffic Accident System (TAS). The original source of the data is the MV6020 (was MV104) forms which police officers fill out for each accident in the province.

On the way from the MVB file to the HAS master files, the following tasks must be accomplished:

1. Data must be selected from the MVB records, reducing each record to a HAS record.
2. Extract the "ON" and "AT" text information from the MVB records, for storage in a separate file.
3. Fatal Accidents must be separated for special treatment.
4. Validity checks must be done on the accident location codes. This is because they indicate the accident location, so are essential for locating accident-prone highway sections and locations. The location codes are also frequently mis-coded. Furthermore, the eventual existence of a valid location code is used to determine the jurisdiction of the accident location (Rural/Municipal or Provincial), because the jurisdiction code on the MV-104 form is so unreliably coded.
5. Accidents which occurred on Municipal/Rural jurisdiction roads must be separated from those on Provincial roads, because the Highway Accident System deals with Provincial-jurisdiction roads only.

The most labour-intensive part of the update procedure is the corrections which must be done to the records found to be in error.

There are two optional checks which can be done on the data: Location/Landmark Types and Police Code/Segment consistency. These checks have proved impractical, and are not done.

7.2 Selection of Accident Data from TAS

Until mid-2003, accidents were selected by accident date. An update period end date was decided upon (e.g. the end of a quarter), then 6 months were after that date were allowed to pass, to ensure that all accident reports for the period were in TAS, then the accidents for that period were requested, and processed into HAS. If any accident reports for that period should subsequently be entered into TAS, they would never make it into HAS.

In the summer of 2003, in order to keep HAS more up-to-date, the HAS update system was changed to get accident data from TAS by the TAS data entry date.

The transition update was 200307. Accidents for this update were selected from TAS with: accident_date >= 1-jan-2003 AND data_entry_date <= 31-Jul-2003. In subsequent updates, data will be selected simply by data_entry_date.

7.3 General Information

7.3.1 Workstation (PC) Folders

Create a separate folder on the PC to contain the job output, work files and notes for each update.

On the Cypher workstation, the update folders are currently (2001-07-17) under `c:\has\upd` and are named yyyymm, where mm is the last month of the update period. eg `c:\has\upd\200103` for the year 2001 first quarter update.

7.3.2 Submitting Jobs

To submit a Highway Accident System Update job, select the Update option on the HAS main menu, to get the Update Main Menu:

```
UPD           Highway Accident System  -  Update Main Menu
              -----------------------     ----------------

            Current Update:  200006  (Year and Month: yyyymm)

               1.  Submit Update jobs
               2.  Edit the Fatal accident Error file
               3.  Edit Non-fatal accident Error file

               4.  Start a New Update
               5.  Change to a Different Update

               R   Review the Update job log

               X   Return to previous menu
               Z   Exit to TSO

    Option  => 1
 New Update =>            (yyyymm, for options 4 and 5)

     (An Update must include ALL the data of the specified month.)
```

Start or change to the desired update. (It is possible to work on multiple updates concurrently.) Then select option 1 (Submit Update Jobs) to bring up the Update Job Submission panel:

```
USEL         Highway Accident System  -  Update Job Submission
             -----------------------     --------------------
                     Update:  200006   (Year, Month: yyyymm)

  Job    Runs    Description
  ---    ----    --------------------------------------------------
  U00      1 S   Setup for a new update
  U01      1 S   Copy data from the MVB file
  U05      2 S   Create the HSBFATAL file
  U10      1 S   Extract the Fatal accidents, and check Fatals
  U15            Put the Fatal edits into effect
  U20      1 S   Check the Non-fatals
  U25            Put the Non-Fatal edits into effect
  U30            Merge Fatals and Non-fatals; divide by jurisdiction
  U38            Merge new LOCTEXT data to create new LOCTEXT.MASTER.
  U40            Update the Current master files, and create backup files.
  U45            Move old data from Current to Archive files
  U90            Clean Up Disk Files

   R     - Review the Update Job Log
   X     - Exit to previous menu (Z to exit to TSO)

 >           (NUMBER OF JOB TO SUBMIT, R, X or Z)
```

The Update Job Submission Panel lists the job names, the number of times each job has already been submitted during THIS Update, a job completion indicator, and a description of each job. The Job Log can be reviewed from this menu (option R). Enter X to return to the main menu, and enter Z to re exit from HAS.

Enter the 2 digit number of the Update job name (e.g. '10' for job U10) and press ENTER. If the job has been submitted previously for this update, the Job Log records for the previous runs will be displayed.

Depending upon the job being submitted, requests for other information may appear. For example, for job U45, which moves data from Current to Archive files, the oldest year of data to remain on the Current file is prompted for.

The Job Submission Confirmation panel will appear as for Data Retrieval job submission.

The job completion indicator is a single character which appears to the right of the number of runs.
When a job is submitted it remains blank. When the job has completed, it is:
S - if it completed successfully,
X - if it did not complete successfully.

Note: The job completion indicator is obtained by the system by reading the Job Log file. The Job Log file is read each time the Update Job Submission screen is re-displayed. Leaving the option field blank and pressing Enter will cause the screen to re-display. Therefore, you can submit a job, and if you expect it to run quite quickly, you can just press Enter periodically, until an S or X appears next to the job number.

7.3.3 Checking Job Output

The following notes in this section apply to all jobs in general.

The last page of every job, should contain two printed lines. The first should be as follows:

```
=== JOB LOG RECORD PRODUCED BY PROGRAM THAS071 ===
```

The second line should be the Job Log job completion record for the job. (See next section.)

Make sure that this last page is present, and make sure that the job log record ends with 'COMPLETED SUCCESSFULLY'. If it ends with an error message, or if the last page of the listing is not as described here, do not run any more Update jobs until the problem has been sorted out.

The output should also be checked for error or warning messages, and for reasonable record counts. When looking at the job output online (in IOF), record counts can be found by searching for the string SUMMARY.

Most jobs have one or more steps which execute the system utility program IDCAMS. The IDCAMS step will often complete with a completion code of 8. In the IDCAMS output, there will often be messages saying that one or more files could not be found, and were not deleted. This is all normal.

Snap the job out of IOF, and download it to the workstation.
Eg for job U10, if it was job 4 in the IOF job list:

```
IOF:        SD DS(JU10.JOB)
            4 SNAP
            SNAPCLOS

PC: (using the Cypher ft.bat ftp utility)
            ft get f ju10.job

PC: (using ftp directly)
            ftp bcsc01.gov.bc.ca
            userid
            password
            get ju10.job
            quit
```

After downloading, delete JU10.JOB on the mainframe, with the TSO command:
`DELETE JU10.JOB`

7.3.4 Job Log File

For each update, a log of all the Update jobs run is automatically maintained in a file. The file is called THASP.UPyyyymm.JOBLOG

There are normally two log records for each job run. The first one (which is put on the file by the TSO EXEC program THASJRUN) is the job submission record. It contains the following fields:

- Date and time of the job submission
- 'JOB'
- the three character job name (eg 'U80')
- the submission number of this job: '04' if this is the fourth submission of this job in this update.
- the job name, as appears on the listing banner page. For job U80 this will be 'THASPU80', unless USERID job names have been requested (see the description of the Parameters panel).
- the job number, as appears on the listing banner page.
- the user ID under which the job was submitted
- 'SUBMITTED'

When the job ends, a completion record is written to the job log, and to the end of the job listing. This record is the same as the submission record with the following exceptions:

the date and time are the date and time the job completed
the job number and userid are omitted
instead of 'SUBMITTED', the record will end with:

'COMPLETED SUCCESSFULLY'	- if all went well
'ERROR: COND > 8'	- if any step in the job had a completion code greater than 8
'TERMINATED ABNORMALLY'	- if an error occurred which caused a system ABEND.

If there was a JCL error in the job, no completion record is produced - on the listing or in the job log.

Example Job Log File

```
   Menu  Utilities  Compilers  Help
 ------------------------------------------------------------------------------
-
 BROWSE    THASP.UP200006.JOBLOG                      Line 00000000 Col 001 080
 Command ===>                                              Scroll ===> PAGE
******************************** Top of Data
********************************
2001-05-31 14:53 JOB U00 01 HAS0JU00 03019 SC81171 SUBMITTED
2001-05-31 14:53 JOB U00 01 HAS0JU00               COMPLETED SUCCESSFULLY
2001-05-31 14:55 JOB U01 01 HAS0JU01 03042 SC81171 SUBMITTED
2001-05-31 14:55 JOB U01 01 HAS0JU01               COMPLETED SUCCESSFULLY
2001-05-31 14:57 JOB U05 01 HAS0JU05 03073 SC81171 SUBMITTED
2001-05-31 14:57 JOB U05 01 HAS0JU05               COMPLETED SUCCESSFULLY
2001-05-31 15:28 JOB U05 02 HAS0JU05 03489 SC81171 SUBMITTED
2001-05-31 15:28 JOB U05 02 HAS0JU05               COMPLETED SUCCESSFULLY
2001-05-31 15:29 JOB U05 02 HAS0JU05               COMPLETED SUCCESSFULLY
2001-05-31 17:44 JOB U10 01 HAS0JU10 05072 SC81171 SUBMITTED
2001-05-31 17:45 JOB U10 01 HAS0JU10               COMPLETED SUCCESSFULLY
2001-05-31 17:47 JOB U20 01 HAS0JU20 05112 SC81171 SUBMITTED
2001-05-31 17:49 JOB U20 01 HAS0JU20               COMPLETED SUCCESSFULLY
******************************** Bottom of Data
********************************
```

7.3.5 LKI File Requirements

The H.A.S. Update system requires the following files for verifying Location Codes:

Note that the versions of these file which were in effect at the time the accident locations were coded should be used! E.g. if an accident was coded at time t1, and the segment was modified effective time t2, then the THASP.SEGMENT file for time t1 should be used. This is not handled automatically by the system. It must be managed manually by modifying THASP.PROCLIB(THAS030).

```
THASP.SEGMENT     - Segment file

THASP.HIGHWAY     - Highway file

THASP.SHNFIL      - Segment/Highway/Node file, created by job PDSM
                  - this is used by THAS030/THASSRY/THASHWY
```

The following files may be needed, depending upon options specified in your Parameter menu. Whether or not these files are used, they must at least exist.

```
THASP.SEGDTCH     - Segment/Police Detachment file.
                  - may be used used by the Update sub-system to
                    verify that segment and police codes match.

THASP.LANDMARK    - Landmark file
                  - may be used by the Update sub-system to verify
                    that Location Type coded on the MV104 is
                    equivalent to the Landmark type.

THASP.LMATCH      - Location Type - Landmark Code equivalences
```

These files are defined in detail in the Files section of this manual.

7.3.6 File Names, Flow Charts and Job Descriptions

All files created for an update have THASP.UPyyyymm as their first two index levels, where yyyymm is the update date specified on the Update Main Menu.

To understand how and where each update file is created, reference the job flowcharts. Also, see the **Update Job Descriptions** sections for a written description of each job.

7.3.7 How the Location Codes are Checked

The Location Code is composed of three sub-fields - Highway, Segment and KMMARK (distance).

Program THAS030, in jobs U10 and U20, performs the following checks on the Location Code:

- For non-fatal accidents, if the location code is blank, no further checking is done, and the record is diverted so that it does not appear on the error file.
- The Highway number and letter must be on the LKI Highway File.
- The Segment number must be on the LKI Segment File.
- The Highway-Segment combination must be on the LKI Highway/Segment file.

```
- The KMMARK must be numeric and positive, and must be a valid
  distance in the segment.
  (0 <= KMMARK <= segment length in the LKI Segment file.)
```

7.3.8 Downloading and Uploading Files

The HAS Update process now involves file processing done on the Windows workstation.
Files can be up and downloaded to and from the mainframe using the command-line FTP which comes with Windows. On Cypher Consulting workstations, there is a utility called FT which automates some of the FTP commands. For details on how to use FT, type FT at a Windows command line on a Cypher Consulting workstation.

It is important to note that you cannot FTP directly into a mainframe dataset with the THASP high level index. This is because of RACF security: the mainframe FTP program assumes your default RACF group, THASD, which does not allow modification of THASP datasets. There is no way to change the RACF group in FTP. Therefore, to upload into a THASP dataset, you have to upload to either a THASD or a personal (<userid>) dataset, then rename or copy the file in TSO / PDF.

Example Download

Using FTP

```
ftp bcsc01.gov.bc.ca
userid
password
get 'THASP.UP200101.CSVFATAL' fatal.csv
quit
```

Using FT

```
ft get f 'THASP.UP200101.CSVFATAL'
rename 'THASP.UP200101.CSVFATAL'  fatal.csv
```

Example Upload

Using FTP

```
ftp bcsc01.gov.bc.ca
userid
password
put NonFat.edit
quit
```

Using FT

```
ft put f nonfat.edit
```

In TSO/PDF Option 3.3 (Move/Copy)

```
Command> C
From Dataset> nonfat.edit
(Press Enter)
To Dataset>   'THASP.UP200101.NONFAT.EDIT'
```

Finally, delete the intermediate file at a TSO prompt:

```
DELETE nonfat.edit
```

7.4 Preparation for an Update

7.4.1 Start the Update on the Mainframe

Select option 4 (Start a New Update) on the Update Main Menu, and enter the year and month of the last month of the update period, in YYYYMM format, on the New Date line. When you press Enter, the new date replaces the date at the top of the screen.

This creates the Job Log file for the new update.
It also allocates the THASP.UPyyyymm.MVB file, and permits it to Wayne Meckle of ICBC, so that he can upload data into it.

7.4.2 Initialize Generation Data Groups - Job U00

Get to the Update Job Submission menu by selecting option 1 on the Update Main Menu.

Submit job U00.

This job sets up the required Generation Data Groups for the update.

7.4.3 Email Request for Accident Data

The MVB contact for accident data is:

Wayne Meckle, ICBC
wayne.meckle@icbc.com
250-414-7925

Email Wayne: send him the new dataset name, and ask for accident data with a TAS data entry date in the desired period. **Note**: Accident data is entered into TAS, then loaded into the TAS Data Warehouse weekly (on Sunday). The data for HAS comes from the data warehouse. Therefore, **do not ask for data until after the first Sunday of the month**. (Wayne Meckle would probably wait, but he might forget, or he might be replaced by someone who would not wait!)
Example:

```
Hello Wayne,
Could you please extract the accident records which
were entered into TAS during November 2004, and upload
them to the mainframe dataset 'THASP.UP200411.MVB'.

Thanks.
```

7.4.4 On The Workstation

- Create folder `has\upd\yyyymm`
- Copy in GETFAT.BAT and PUTFAT.BAT from folder \\CYPHER_XP\C_XP\HAS\FATALS, (or from the previous update directory).

7.4.5 Set Parameters

Use the HAS Parameters panel to set any parameters necessary. Normally, no changes are required.

7.4.6 Check LKI File versions in Proc THAS030

Check the versions of the LKI files used by program THAS030 in Proc THAS030. The versions used must be those in effect at the time the accidents occurred. If this is not possible (e.g. because the accident data times span a change date), you must be aware of the LKI changes and their effective dates when doing location code editing. If necessary, use the "V" edit option to force through a location code which is invalid according to the LKI files being used, but was valid at the time of the accident.

7.4.7 Check the HSB Fatal File (obsolete)

This step is now obsolete. When the HAS was designed, the Highway Safety Branch used to get all fatal accident information on paper before getting it electronically from the MVB. Their location codes were more reliable than those on the MVB records. The data was compiled in a file called the HSBFatal file, which is described in the Files section of this manual. The file was compared with the MVB data (in job U10), and taken as the authoritive fatal accident data.

This process has now been short-circuited - the HSBFatal file is created from the electronic data in job JU05.

7.5 Create an Accident Data File from the MVB File - Job U01

When Wayne Meckle confirms that he has loaded the data into file THASP.UPyyyymm.MVB, submit job U01.

Check the job listing:

- the job checks that multi-page accidents have matching key fields, and that the page numbers ascend sequentially. If there are any related error messages, it means that there is an error either in program THAS010 or in the MVB data.
- check the accident and record counters in the PROGRAM THAS010 SUMMARY, to make sure they look reasonable.
- check that the date ranges of the data found look reasonable. (Now that accidents are selected by TAS data entry date instead of accident date, the range of the accident dates will vary.)
- check for duplicate CASE messages. A duplicate case+date is probably an updated record. A duplicate case only could also be an update (if the date was changed) or could simply be a duplicate case number. See notes below:

Checking duplicate accident records:

- THASD.UTILJCL(RUN088) can be used to extract specified accidents, e.g. from HSBACC files.
- RUN088 once for each of the new and old files.
- then run THASD.UTILJCL(RUN916) to compare the two accident files.
- If it is the same accident, assume the new one is a replacement: Before or after running job U40, edit the latest generation of THASP.PROV.VALLOC.CURRENT and manually delete the old one. (THAS951 could be simplified to do this delete job, but the manual edit is easy.)

Note that with the original method of getting accidents from ICBC by accident date, with a lag of 6 months, this update situation was not an issue. As it is, we are probably only getting accident *replacements*, not updates, since we now select by Entry Date. This whole issue should be addressed in the system rewrite.

7.6 Process Fatal Accidents

In the original design of the system, it was assumed that the fatal accidents would all be processed before work began on the non-fatal accidents. However it may be desirable to work on the non-fatals while waiting for information to get the fatals done. To accomplish this, run job U10 after job U05, to create the NONFAT file, then proceed to job U20. Job U10 will then be re-run when the fatal accidents have been verified.

7.6.1 Create the HSBFATAL and CSVFATAL Files - Job U05

Submit job U05.

7.6.2 Verification of Fatal Location Codes

For invalid or suspicious location codes, make a list of case numbers and dates, and send to Wayne Meckle (ICBC), asking for police file numbers. (This information is stored in a ICBC database separate from TAS.)

When armed with the police file numbers, phone the local police detachment, and ask for details. Police contact information is maintained in HASLKI_DATA.MDB, table Police_Addresses.

(This work used to be done in the Highway Safety Branch in Victoria. After staff reductions, Brad Blaney tried to carry on, but could not keep up. So we decided to devolve the job to the regions. They did not all get the work done in a timely manner, and the HAS updates were still falling behind, so we gave up on the principle that all fatal accident locations should be verified.)

Download the Fatal Data

Make sure you have a SPAN connection. (If not, type SPAN at a Windows command prompt.)

Run GETFAT.BAT at the Windows command prompt, in the update folder, . (No arguments required.)

This will download 'THASP.UPyyyymm.CSVFATAL' and process it, producing file FATALS.CSV on the workstation.

Verify the Fatal Data

Open FATALS.CSV with MS-Excel, and save as FATALS.XLS - your working copy of the file.

The objective is to get the correct data in the the Hwy, Seg, Km, and Jur columns.
We are only interested in the Jurisdiction 1 (Provincial) accidents, so first make sure the Jurisdiction is correct.

Then for Jur 1 accidents: verify Hwy Seg & Km and correct as required. Use the other data provided, and if that is not sufficient, phone the police detachment which coded the accident. (Police contact information is maintained in the HASLKI database Police_Addresses table.)
Add explanations (e.g. information sources) in the Notes column.

DO NOT CHANGE the Case or MVB fields. Change the Hwy, Seg, Km,Jur and Notes columns only.

Upload the Verified Fatal Data

In Excel, save the `fatals.xls` file as `fatals_verified.csv`

Make sure you have a SPAN connection. (If not, type SPAN at a Windows command prompt.)

Run PUTFAT.BAT at the Windows command prompt. (no arguments required). This will reformat the data from `fatals_verified.csv` into HSBFATAL.TXT, then upload that file to the mainframe (as <userid>.HSBFATAL.TXT).

On the mainframe, copy HSBFATAL.TXT to 'THASP.UPyyyymm.HSBFATAL', as follows:
(Mainframe security - RACF - does not allow you to ftp directly to a THASP dataset.)

- log on to TSO, and go to PDF option 3.3 (Copy)
- specify C to copy, in the Command field.
- specify From dataset: `HSBFATAL.TXT`

- specify To dataset: 'THASP.UPyyyymm.HSBFATAL'
- (a message will appear warning that truncation may occur. This is OK.)

7.6.3 Extract the Fatal Accidents - Job U10

Make sure that the HSBFATAL file has been prepared as described in the previous section.

Submit job U10.

Job U10 matches the HSBFATAL file against the accident file, and divides the fatal accidents into separate files. If Location or Jurisdiction codes on matching records differ, the HSBFATAL file information is used. It also checks how well the HSBFATAL file and the accident file correspond, using the TOTALKLD field of the accident file.

The job then checks the validity of all the Jurisdiction 1 Fatal accident Location Codes.

Check the job U10 listing:

Check the job log record at the end of the listing.

Check the PROGRAM THAS020 Output:

- check the record and accident counts in the summary
- Check over the messages which precede the summary: especially for messages about TOTALKLD > 0 BUT NO RECORD ON HSBFATAL FILE. These messages indicate that fatal records got lost in the HSBFATAL / regional file shuffle!
- check the mismatch statistics which follow the record counts. If there were any mismatches, look for details in messages preceding the THAS020 SUMMARY.

Check the PROGRAM THAS030 SUMMARY:

- see if any records were written to the Invalid Location file. If so, there are corrections to be made.

PROGRAM THAS050 creates an edit file if there were any records written to the Invalid Location file by THAS030. The THAS050 record counts should show that one edit record was written for each accident on the Invalid Location file.

- a printed copy of the edit file follows, on the second to last page of the listing.

If the job listing indicates that there were coding errors or omissions in the HSBFATAL file, they must be corrected as described in the following section.

7.6.4 Correcting Fatal File Errors

Case 1:

If there were omissions or jurisdiction code errors in the HSBFATAL file, fix the HSBFATAL file, and re-run job U10. (If there were also Location Code errors, fix them on the HSBFATAL file as well, before re-running job U10.)

Case 2:

If there were no omissions in the HSBFATAL file, but there were Location Code errors, they must be corrected in the HSBFATAL file.
You can then either:

- re-run job U10, or
- make the same Location Code corrections in the Edit File (as described in the following section) then run job U15.

Case 3:

If Police and Location Type code checking is selected, and there are Police or Location Type errors, and all HSBFATAL file omissions have been corrected as described in Case 1, fix the Police and Location Type errors in the Edit file, then run job U15.

7.6.5 Edit of the Fatal Edit File

The Fatal Edit File can be edited on the mainframe using an option of the Update Main Menu, or by editing THASP.UPyyyymm.FATJUR1.EDIT from PDF. Alternatively, the edit file can be downloaded, edited on the PC, then uploaded.

The Fatal Edit file is edited in the same manner as the Non-fatal Edit file. See the relevant sections on non-fatal file processing for details.

If you wish to remove a record from the Fatal file (i.e. if you decide it was not a fatal accident after all), or if you wish to change the Jurisdiction code, you must either delete the record from the HSBFATAL file or change the Jurisdiction code on the HSBFATAL file, and start back at job U10. (If non-fatal processing has already begun, the now non-fatal accident will have to be moved over into the non-fatal stream. There are no established procedures for this as yet.)

7.6.6 Put the Fatal Edits into Effect - Job U15

Job U15 applies the edits done to the Fatal Edit file to the corresponding accident file which has the errors (file THASP.UPyymmdd.FATJUR1.TOEDIT(0)), then re-checks the location codes, creating a new Edit file.

In the job U15 listing, the PROGRAM THAS054 SUMMARY contains the record counts for the edit application step. The rest of the listing should be similar to the job U10 listing, with the THAS030 SUMMARY telling you (with the INVLOC record count) whether any invalid location codes still were found.

Edit file corrections and job U15 submissions must continue until no invalid location codes were found, and the Fatal Edit file is thus empty.

If at any time you wish to look at the previous Fatal Edit file, it is always named:

THASP.UPyyyymm.FATJUR1.EDIT.BAK

==> Do NOT run Job U15 if Job U10 wrote nothing to the Edit file, i.e. found no invalid location codes.

7.7 Process Non-Fatal Accidents - Jobs U20 and U25

7.7.1 Check the Non-Fatal Accidents - Job U20

Job U20 uses the NONFAT file produced by job U10, so if running job U20 before job U10 has been run for the last time, be aware that if subsequent runs of Job U10 put different accidents into the NONFAT file, there is a danger of accidents being duplicated or omitted.

Submit job U20.

This job checks the location codes of the non-fatal accident records. An Edit File is created containing one record for each accident in error. (Note that it is one record per ACCIDENT, not per RECORD - only one Edit record is produced for a multi-page accident.)

Accidents coded with jurisdiction 2 or 3 will appear on the edit file only if they have a non-blank, invalid location code. Accidents coded with jurisdiction 1 with blank location codes will appear on the Edit File.

Check the job U20 listing:

- the PROGRAM THAS030 SUMMARY tells you how many records were in error, and how many of what kind of error were found.
- the PROGRAM THAS050 - EDIT RECORD COUNTS BY POLICE CODE report tells you how many errors were found for each police code, and in each police subdivision.

- the PROGRAM THAS050 SUMMARY tells you how many records were written to the Edit File.

Download 'THASP.UPyyyymm.NONFAT.EDIT'

```
ft get f 'THASP.UPyyyymm.NONFAT.EDIT'                - keep this original
copy 'THASP.UPyyyymm.NONFAT.EDIT'  nonfat.edit       - working copy
```

7.7.2 Description of the Edit File

The Edit records are sorted by Police Code.

The Edit record is divided in half by two vertical bars. The data to the left of the vertical bars is the edit area. The data to the right of the vertical bars is for information only. **NEVER MODIFY DATA TO THE RIGHT OF THE VERTICAL BARS**.

Be sure to keep Highway number and letter in their correct columns.

Edit Area Description: (left of the vertical bars)

```
cols.  3- 6      Police Code
                 Location Code
cols.  8-10        Highway Number
col.  11           Highway Letter
cols. 13-16        Segment Number
cols. 18-21        Kmmark
cols. 23-24      Location Type
```

An asterisk following a field indicates that field is in error.

A cross hatch (#) between the Highway and Segment fields indicates that the Highway and Segment fields are both valid in themselves, but the Segment is not in the Highway, according to the LKI file THASP.HWYSEG.

The location code fields are separated by underscores if there are no error indicators.

Parse Flag: (between the vertical bars)

V - this indicates that the Location Code was originally invalid, but was modified and made valid by program THAS030, by examining (parsing) the location code as originally keypunched and by looking at the LKI files.

\- the record was put on the Edit File so that the user can verify that the 'correction' was done correctly. (See Action Code 'C' in the following section.)

X - this indicates that the Location Code was modified by program THAS030 in an attempt to fix it, but is still invalid.

Information Area (right of the vertical bars - DO NOT MODIFY)

```
MVB Location Code        - as keypunched from the MV104
Jurisdiction Code
Accident Case Number
Accident Date
Place (city)
ON: AT: text from the MV104.
```

Example Edit File (On/At information truncated)

```
0101  97A_    *    *02 | | 97A        1 R2763305 20000509 SPALLUMCHE ON: HWY 97A ( VERNON -
0101  97A_    *    *02 | | 97A        1 R2763347 20000528 SPALLUMCHE ON: HWY 97A ( VERNON -
```

```
0101  97A_1122_    *02 | | 97A1122     1 R2775404 20000512 ARMSTRONG  ON: HWY 97A AT: 3 KMS
0101  97A_1122_0166 03 |V| 97N11220166 1 R3097740 20000607 ARMSTRONG  ON: HWY 97A AT: 4236
0101  97A#1184_4344 02 | | 97A11844344 1 R2793205 20000428 ARMSTRONG  ON: HWY 97A AT: REST AREA
0102  97 #1122_0261 01 | | 09711220261 1 R3101364 20000418 ENDERBY    ON: GEORGE AT: KING
0102  97 #1122_0265 01 | | 09711220265 1 R2732693 20000405 ENDERBY    ON: HWY 97 AT: CLIFF AVE
0102  97A_1122_0336 14 |V| 97A1122 336 1 R2732818 20000605 ENDERBY    ON: HWY 97A AT:
0104  97 _    *    *02 | | 097         1 R3064981 20000404 LAKE CNTY  ON: HWY 97 AT: NORTH OF
0104  98 *    *    *01 | | 098         1 R3064919 20000623 WESTBANK   ON: ??? AT: BROWN ROAD
0104  97 _0062*4000 02 | | 09700624000 1 R3064423 20000604 PEACHLAND  ON: HWY #97 SOUTH AT:
0104  97 _1115_    *02 | | 0971115     1 R2730668 20000505 PEACHLAND  ON: HWY 97 AT: 2 KMS
0104  97 _1115_0470 13 |V| 0971115 470 1 R3138951 20000521 PEACHLAND  ON: HWY 97 SOUTH AT: NEAR
0104  97 _1115_0501 01 |V| 0971115 501 1 R3064720 20000610 PEACHLAND  ON: HWY 97 AT: PRINCETON
```

7.7.3 Editing Procedure

Originally, the Edit File was edited on the mainframe using an option of the Update Main Menu, or by editing THASP.UPyyyymm.NONFAT.EDIT from PDF.

The Edit File is now processed in Windows. If not already downloaded (as described in the JU20 section above), download the Edit File to file Nonfat.edit

When editing the file:

DO NOT MODIFY FIELDS TO THE RIGHT OF THE VERTICAL BARS.
DO NOT DELETE OR INSERT ANY EDIT RECORDS.

Correct fields as required in the edit area. See the previous section for a description of the areas, fields and error indicators of the error file.

The following 'Action Codes' may be put in column 1 of an edit record, to give instructions to program THAS054, which interprets the Edit file in job U25:

V - "The record is VALID, despite what program THAS030 says!"
- "Write the record to file **OVRIDE**."
- This could be useful if the error is caused by a problem in the LKI files which you don't want to fix right away, or if the LKI files being used by program THAS030 have been updated since the time of the accident.

C - "The record has been checked."
- This is necessary to get a record with a Parse Flag of 'V' off of the Edit File.
- Optional for other records - you may wish to use it just to indicate to yourself, between sessions, what you have checked.

X - "The Location Code is uncorrectable - write to file **GIVEUP**"
- This is the normal Action Code for jurisdiction 2 and 3 records.

1 - "The Location Code is uncorrectable, change the jurisdiction code to 1 then write to file **GIVEUP**."
- Use this Action Code if jur 2 or 3 is coded, you have reason to believe that it should be jur 1, but cannot correct the Location Code.

2 - "The Location Code is uncorrectable, change the jurisdiction code to 2 then write to file **GIVEUP**."
- Use this Action Code if jur 1 is coded, but you have reason to believe that it is a jurisdiction 2 accident.

3 - "The Location Code is uncorrectable, change the jurisdiction code to 3 then write to file **GIVEUP**."
- Use this Action Code if jur 1 is coded, but you have reason to believe that it is a jurisdiction 3 accident.

It is not necessary to complete all the corrections before running job U25. All uncorrected records will just reappear on the next Edit File.

7.7.4 Put the Non-Fatal Edits into Effect - Job U25

If editing was done in Windows, upload Nonfat.edit to 'THASP.UPyyyymm.NONFAT.EDIT'. (See example in the Download/Upload section above.)

ft put f nonfat.edit
in TSO/PDF 3.3, copy nonfat.edit to 'THASP.UPyyyymm.NONFAT.EDIT'

Job U25 applies the edits done in the Edit file to the corresponding accident file which has the errors (file THASP.UPyymmdd.NONFAT.TOEDIT(0)), then re-checks the location codes, creating a new Edit file.

In the job U25 listing, the PROGRAM THAS054 SUMMARY contains the record counts for the edit application step. The rest of the listing should be similar to the job U20 listing, with the THAS030 SUMMARY telling you (with the INVLOC record count) whether any invalid location codes still were found.

Edit file corrections and job U25 submissions must continue until no invalid location codes were found, and the Edit File is thus empty.

```
===> DO NOT RUN JOB U25 if the Edit file is already empty,
     i.e. if Job U20 did not write anything to the Edit file.
```

If at any time you wish to look at the previous Edit file, it is always named:

```
THASP.UPyymmdd.NONFAT.EDIT.BAK
```

7.7.5 Merge Fatals and Non-Fatals - Job U30

DO NOT RUN THIS JOB UNTIL BOTH THE FATAL AND NON-FATAL EDIT FILES ARE EMPTY.

Submit job U30.

This job creates the four final files of the update:

THASP.UPyyyymm.

JUR1.VALLOC	- Jurisdiction 1 records with valid Location Codes
JUR1.INVLOC	- Non-fatal Jurisdiction 1 records without valid Location Codes
JUR2	- Jurisdiction 2 and 3 records.
JUR1.INVLOC.FATAL	- Fatal Jurisdiction 1 without valid Location Codes

Check the job U30 listing:

Program THAS061 sets the Jurisdiction Code of all non-fatal records with valid Location Codes to 1. The PROGRAM THAS061 SUMMARY tells you how many of those records did not already have a Jurisdiction Code of 1.

Program THAS060 separates all the non-fatal records without valid Location Codes by their Jurisdiction Code. The PROGRAM THAS060 SUMMARY tells you how many of each there were.

Next comes the output of three sort steps. On the third to last line on each page, after the word OUT, is a record count. These are the record counts of the .JUR2, .JUR1.INVLOC, and .JUR1.VALLOC files, in that order.

Next there are three PROGRAM THAS075 SUMMARY pages. These provide statistics for the three final files of the update.

7.8 Update the LOCTEXT Master Files - Job U38

This job (added in October 2003) extracts the location text data (Place, On & At) for the accidents with valid location codes, adds it to the THASP.LOCTEXT.MASTER file, and recreates the LOCTEXT VSAM file.

Check one last time that jobs U00 to U30 completed successfully (look at the Job Log file), that the Fatal and Non-fatal edit files are empty, and that the sum of the record counts of the three output files of job U30 is the same as the record counts of job U01.

Make sure there are no HAS users signed on: run command %WHOSON.
(This is because the production LOCTEXT.VSAM file is deleted and replaced, which should not be done if a user could be using it.)

If all is in order, submit job U38.

7.9 Update the CURRENT Master Files - Job U40

Verify that job U38 has been successfully run.

Determine whether this update contained any accidents whose case numbers and dates duplicate accidents already in the system, and which should replace the corresponding accidents in the system. If so, have the case numbers and dates at hand.

If all is in order, the Master files can be updated - submit job U40. When job U40 is selected for submission, there is an option to specify changes or deletions to be made to the master file. If this option is selected, panel 720 will be displayed. To delete an accident (e.g. if a replacement is in the update file), specify the case number and date, an asterisk in the Page column, and ***DELETE** in the Field column. Other data corrections can be specified if required.

The job U40 listing contains the output of one SORT and three MERGE steps. (A sort is used for the PROV.VALLOC.CURRENT file in case editing has changed a sort key field.) Following each step, there is a THAS075 SUMMARY, which provides statistics of the enlarged master file. The record counts should be the sum of the old CURRENT Master record counts, and the record counts of the Update files:

Update File	CURRENT Master File
.JUR1.VALLOC	.PROV.VALLOC
.JUR1.INVLOC	.PROV.INVLOC
.JUR1.INVLOC.FATAL	.PROV.INVLOC.FATAL
.JUR2	.MUN

The job then creates archive copies of the the first Accident Data file of the Update, the final three files, the HSBFATAL file and the job log file. There will be six pages each containing just "DATA SET UTILITY - GENERATE". Statistics on the archived files are listed in THAS075 SUMMARIES:

```
THASP.UAyyyymm.HSBACC
THASP.UAyyyymm.JUR1.VALLOC
THASP.UAyyyymm.JUR1.INVLOC
THASP.UAyyyymm.JUR1.INVLOC.FATAL
THASP.UAyyyymm.JUR2.
```

The following two archived files are listed:

```
THASP.UAyyyymm.HSBFATAL
THASP.UAyyyymm.JOBLOG
```

These six files are archived so that:

a) the Update could be re-done starting from the HSBACC file,
b) the data source can be checked in trouble-shooting situations
c) an extra form of data backup is available.

If there are any error or warning messages preceding a THAS075 summary, investigate thoroughly. Do not run any more update jobs until things are sorted out.

Next a backup copy of the new version of each of the three master files is created. This produces more "DATASET UTILITY - GENERATE" pages. These backup files are called:

```
THASP.UAyyyymm.PROV.VALLOC.CURRENT
THASP.UAyyyymm.PROV.INVLOC.CURRENT
THASP.UAyyyymm.MUN.CURRENT
THASP.UAyyyymm.LOCTEXT.MASTER
```

Note: Over the years these archived files will accumulate. Every few years, it may be a good idea to delete the older ones, to save storage costs. You can use PDF option 3.4, and enter

DSNAME LEVEL ===> **THASP.UA***

to get a complete list of archived files.

If you have any doubts at all about whether or not job U40 ran properly, contact the Highway Accident System programmer. Data loss is a possible consequence of ignoring problems with the running of this job.

7.10 Archive Old Master Data - Job U45

This job need not be run for every Update. Also, it need not be run as part of the Update sequence - it may be run at any time.

When you want to move data from the CURRENT Master Files to the ARCHIVE Master Files, submit job U45. During the submission procedure, you will be prompted for the oldest year which is to REMAIN on the CURRENT files. For example, if the CURRENT files contained data from years 1980 to 1986, and you wanted to move the 1980 and the 1981 data to the ARCHIVE files, you would enter '82' during the submission procedure. Note that the ARCHIVE-CURRENT break cannot be in mid-year.

Check the job listing:

For each of the file types: PROV.VALLOC, PROV.INVLOC and MUN, there will be a PROGRAM THAS100 SUMMARY followed by SORT program output.

The THAS100 SUMMARY includes record and accident counts for the following:

```
INACC     - the old CURRENT file
OLD       - the data being removed from the CURRENT file
CUR       - data remaining on the 'new' CURRENT  file.
```

The SORT program output contains the record count for the enlarged ARCHIVE file.

Make sure these record counts are reasonable: they should add up, and should be consistent with master file record counts from previous jobs.

7.11 Create the PDS Master - Utility Job PDSM

After the Tape Master files have been updated (jobs U40 & U45), Utility job PDSM must be run to make the updated PROV.VALLOC.CURRENT data available to the Data Retrieval Sub-System.

PDSM is a Utility Job because it is not only run as part of the Update sequence - it may be run at any time.

Job PDSM submission is described in the Utility section.

7.12 Clean Up Disk Files - Job U90

Job U90 deletes all disk files which have THASP.UPyyyymm as the first two index levels, except for the JOBLOG file and the HSBFATAL file.

Make sure that job U40 has been submitted, and completed successfully, before submitting job U90.

Submit job U90.

As mentioned, the job log file is not deleted. This is because job U90 writes to the file, and because it may be useful to reference job log files of previous updates. When you no longer need the file, delete it using PDF option 3.2 as follows (you may want to print it first):

```
OPTION  ===> D

DATA SET NAME  ===> 'THASP.UPyyyymm.JOBLOG'
```

You will be asked to confirm the deletion (to make sure you really want to delete the file).

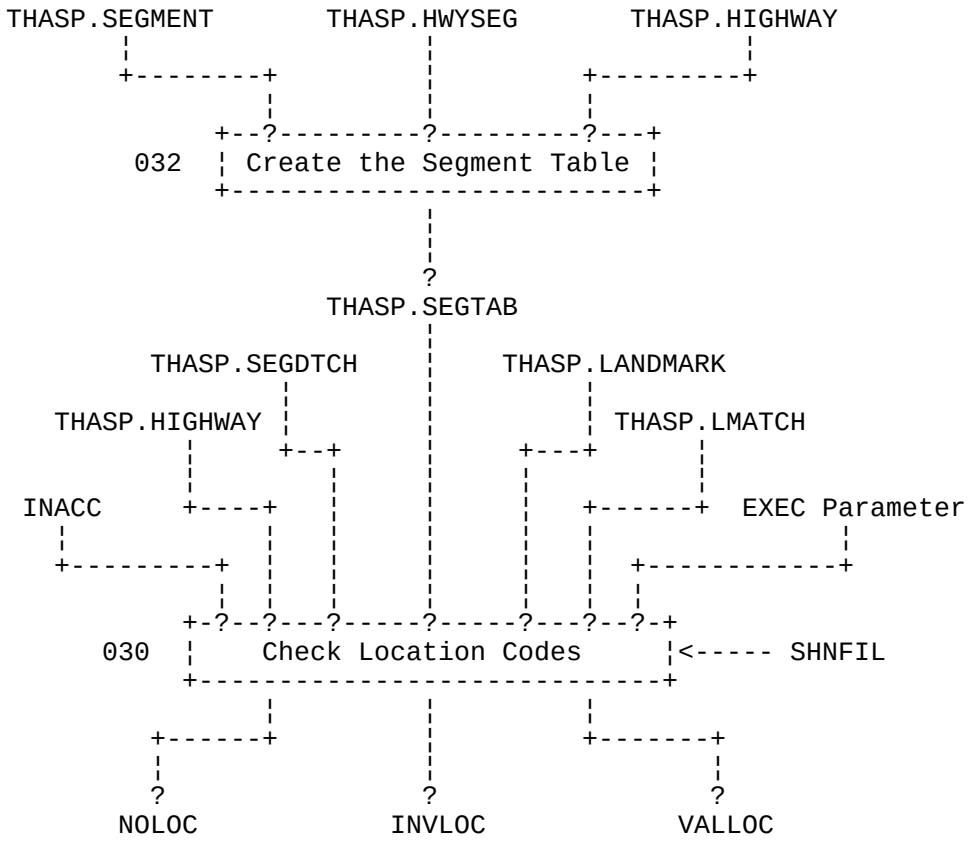
When the HSBFATAL file is no longer needed, it can be deleted in the same manner. The archive version of both these files (THASP.UAyyyymm.*) can be referenced later if necessary.

7.13 Update Job FlowCharts

Interpretation:

- File names are in capital letters.
- Precede file names starting with a period by THASP.UPyyyymm to create the complete file names.
- File names starting with '&&' are temporary files.
- PLI programs are denoted by a short program description in a box. The three digit number to the left of each box is the program number. Precede the program number by THAS to create the complete program name, e.g. THAS030.
- Job steps which use system utilities, such as merge or sort, are enclosed in parentheses, e.g. (Merge)
- Program THAS030 - Check Location Codes - is used in a number of flowcharts/jobs. LKI file detail is not shown each time, but is depicted in the first flowchart.
- Only the primary key of sorts and merges is given.

Detail of Location Code Checking

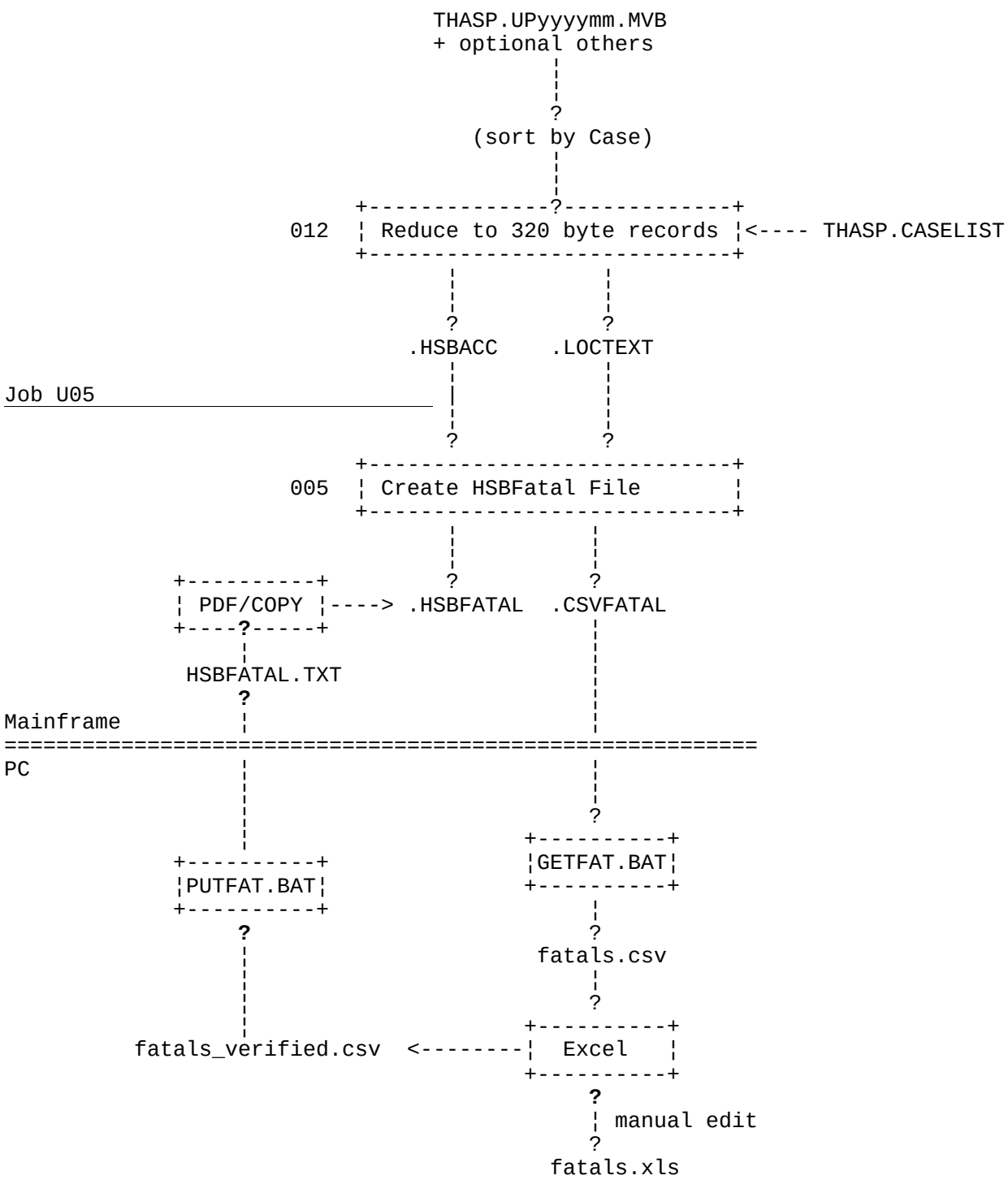


Notes:
- this is the flowchart of the catalogued procedure THAS030, which is used by Jobs U10, U15, U20, and U25
- see the THAS030 program description for interpretation of the DD names INACC, NOLOC, INVLOC and VALLOC, which are parameters sent to the catalogued procedure (differing in different jobs)
- the EXEC parameter has the following values:

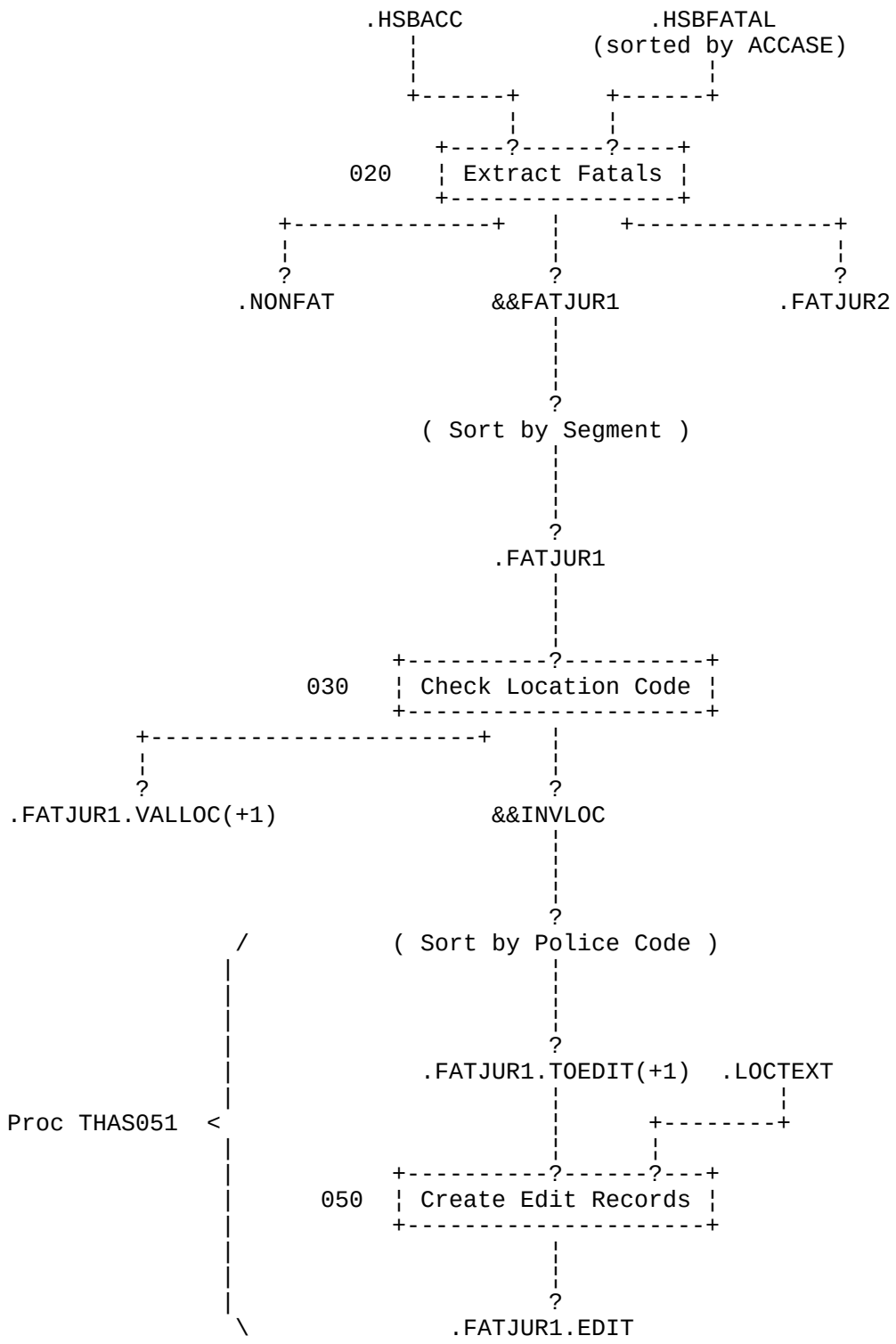
'/BLANK'	- to separate blank location code records into file NOLOC
'/PARSE'	- to parse invalid MVB_LOCN to find valid subfields
'/BLANK PARSE'	- to do both
'/ '	- to do neither

Note: reading both SEGTAB and SHNFIL is redundant, but routine THASSRY uses SHNFIL, thus keeping it independent of PROC THAS030

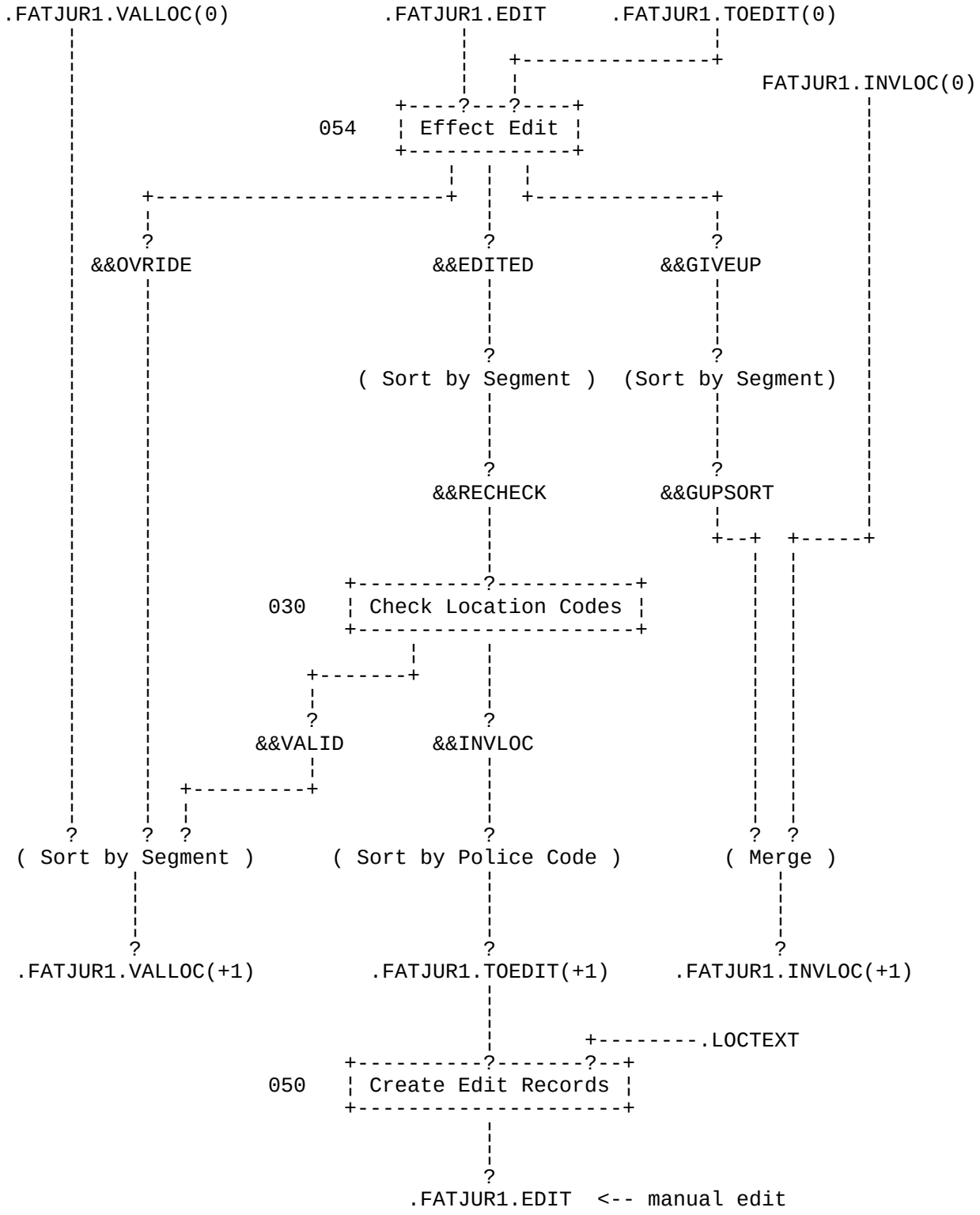
Job U01 Get Data From the MVB File



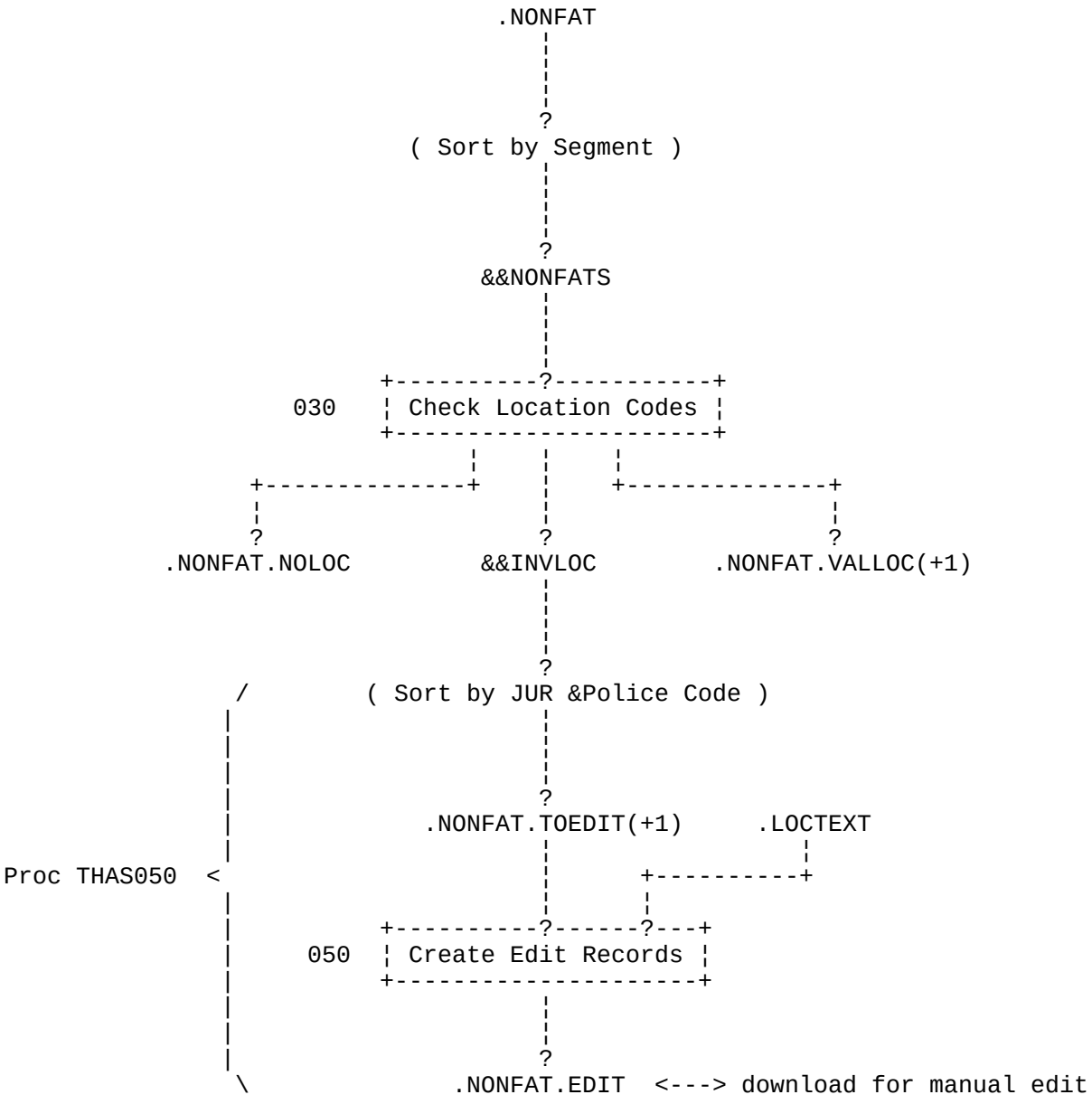
Job U10 Extract Fatal Accident Records



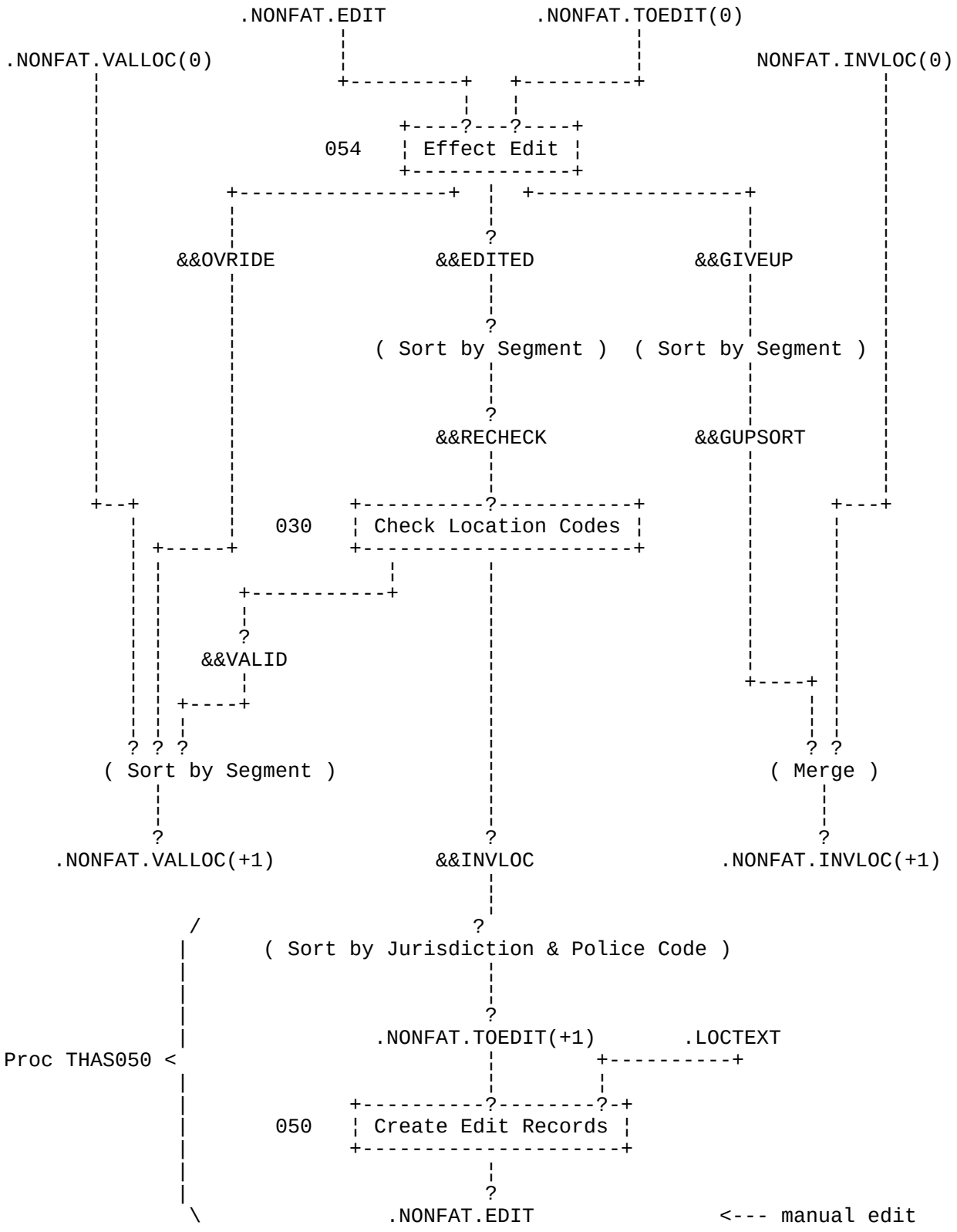
Job U15 Edit and Recheck the Fatal Jurisdiction 1 Records



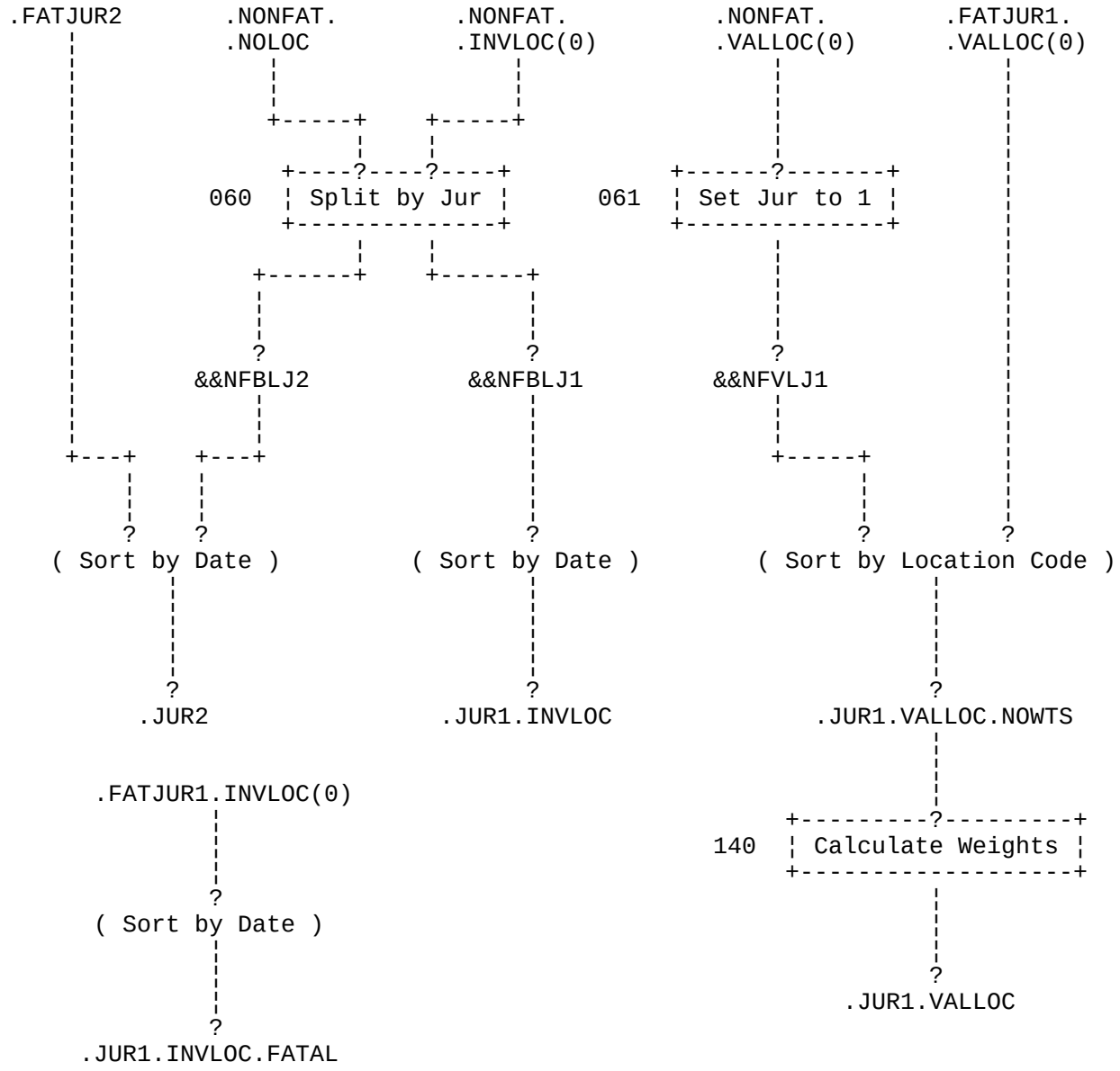
Job U20 Check the Non-Fatal Location Codes



Job U25 Edit and Recheck the Non-Fatal Location Codes

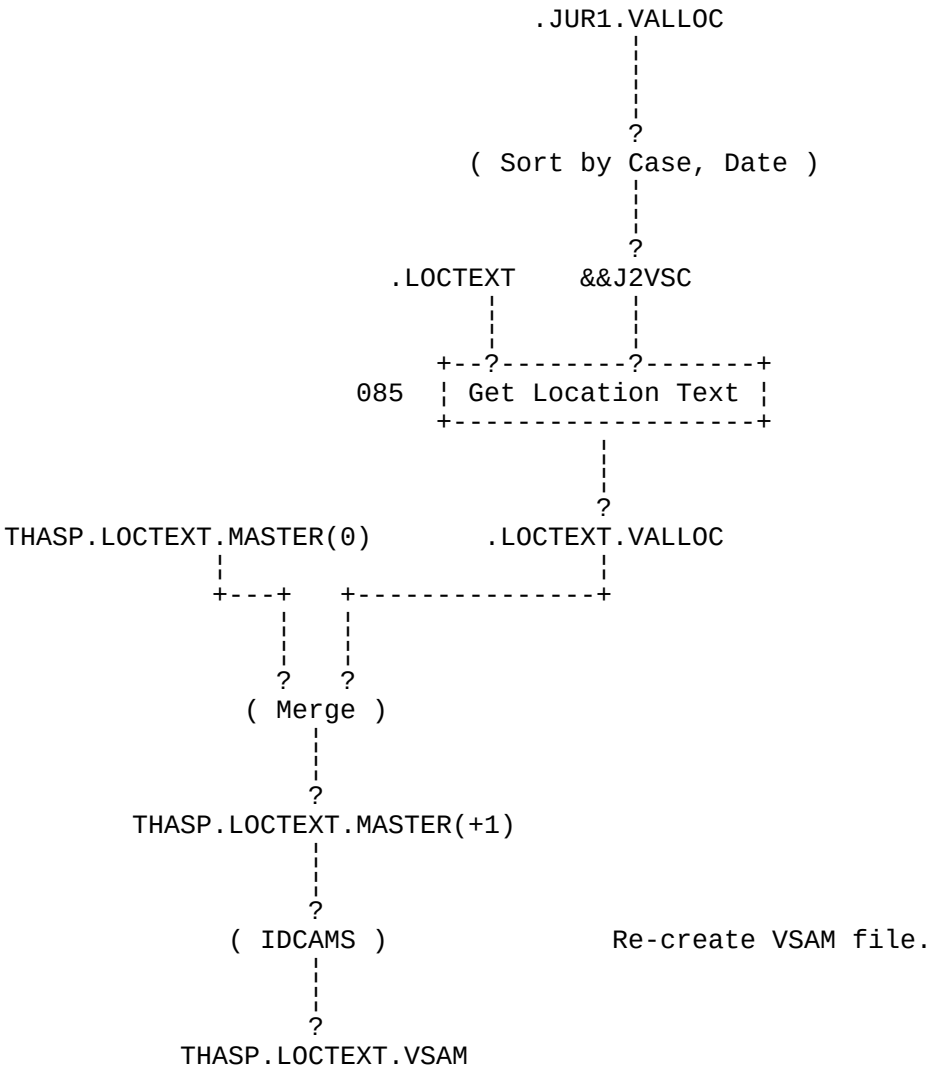


Job U30 Add Fatals, Split by Jurisdiction and Sort by Location Code



Location Text Master Files Update and Archive

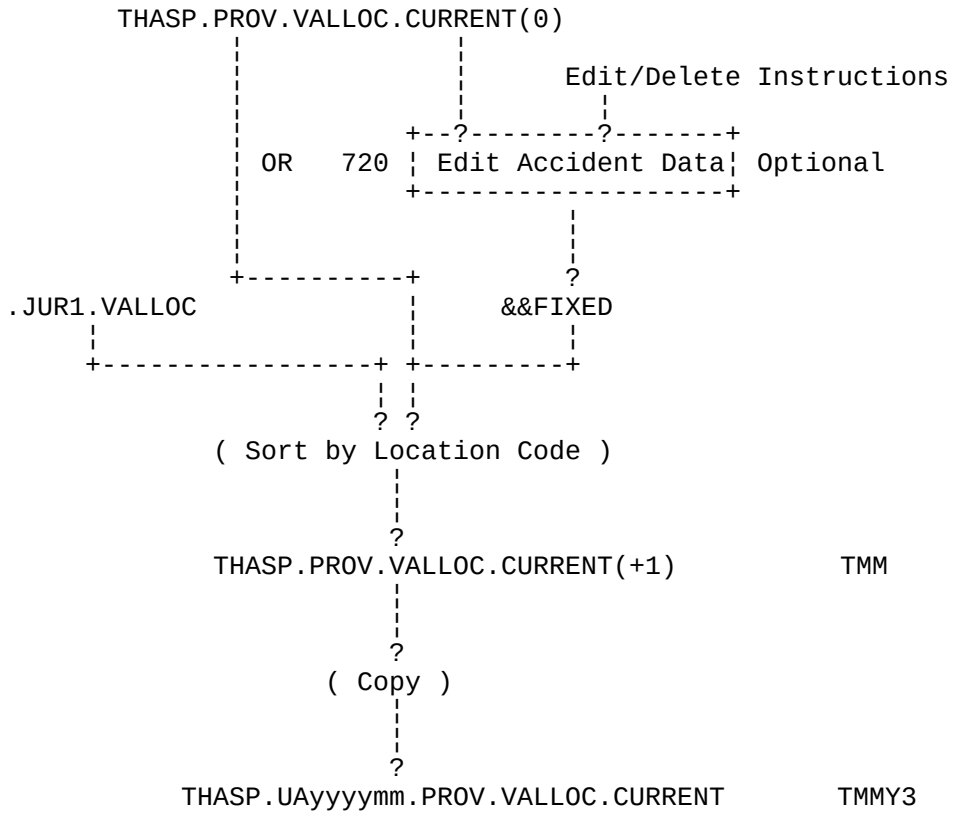
Job U38



Sequential Master Files: Update and Archive

Job U40

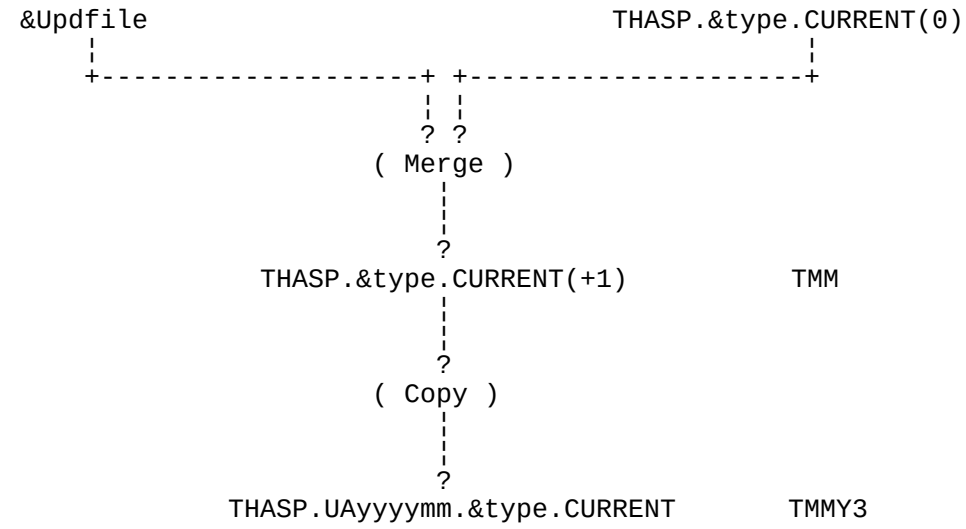
Valid Jur 1 Data



(Job U40 continued on the next page)

Job U40 (continued)

Rejected Data



The process above is repeated three times, with &Updfile and &type replaced by:

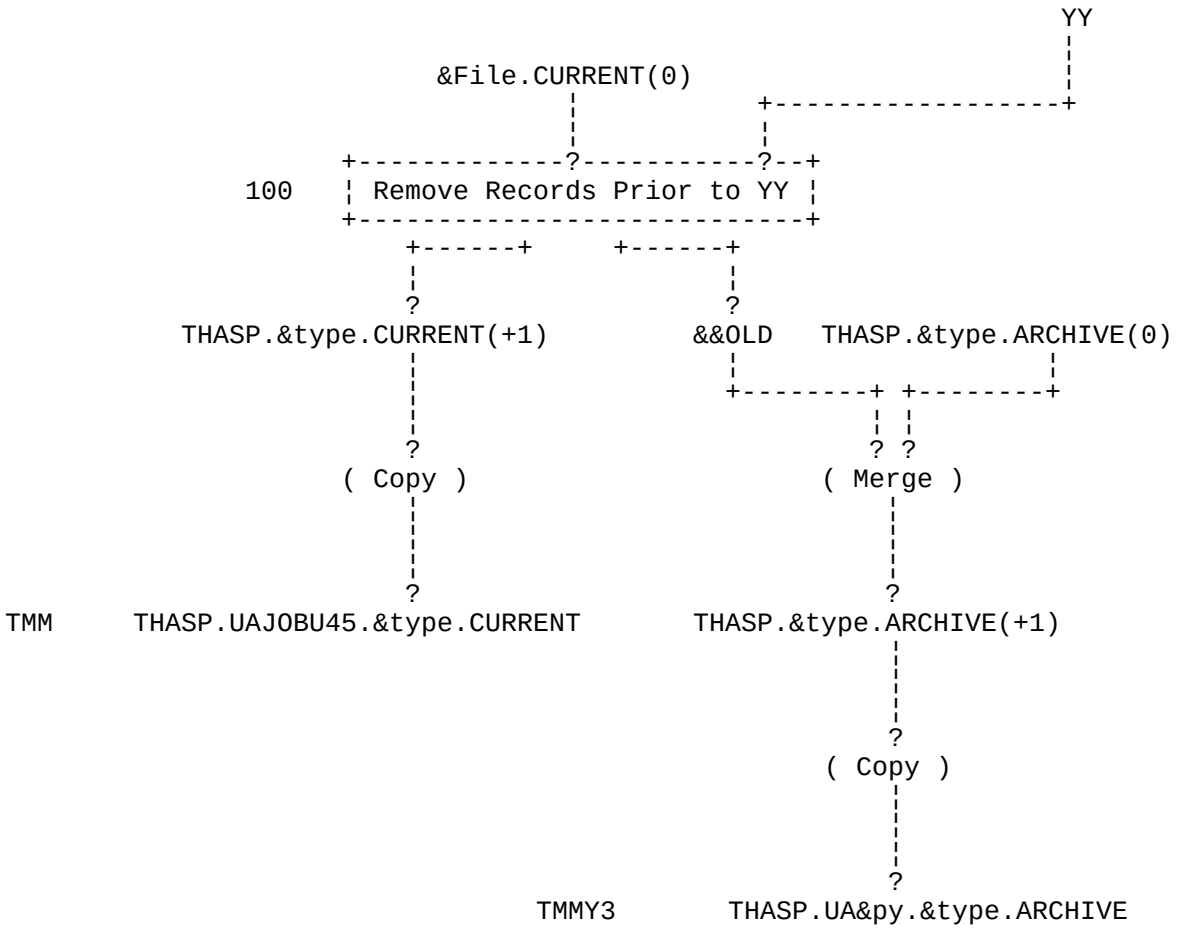
	&Updfile	&type	
1.	THASP.UPyyyymm.JUR1.INVLOC.FATAL	PROV.INVLOC.FATAL	(added 2001)
2	THASP.UPyyyymm.JUR1.INVLOC	PROV.INVLOC	
3	THASP.UPyyyymm.JUR2	MUN	

Then archive copies of Update files are created as follows:

THASP.UPyyyymm.HSBACC	==>	THASP.UAyyyymm.HSBACC
.JUR1.INVLOC.FATAL	==>	.JUR1.INVLOC.FATAL
.JUR1.VALLOC	==>	.JUR1.VALLOC
.JUR1.NOVALLOC	==>	.JUR1.INVLOC
.JUR2	==>	.JUR2
.HSBFATAL	==>	.HSBFATAL
.JOBLOG	==>	.JOBLOG
.LOCTEXT.VALLOC	==>	.LOCTEXT.VALLOC

- where yyyymm is the last month of the Update.

Job U45 ARCHIVE OLD MASTER DATA



Where: YY is the earliest year of data left on the CURRENT files,
&py is the Previous Year - the latest year put onto the ARCHIVE files.

This process is repeated three times, with &type replaced by:

1. PROV.VALLOC
2. PROV.INVLOC
3. MUN

Note:

- that the update step may be done more than once before the archive step is done.
- (+1) files automatically become (0) after a job ends.
- PROV.INVLOC.FATAL is not archived, because it is a small file, and was not created prior to the 199909 update (done in March 2001)

7.14 Update Job Descriptions

Note that the job numbers are not related to the numbers of the programs which they execute.

Job U00 Prepares for a new update - creates an empty Job Log file, and sets up generation data groups.

Job U01 Copies the MVB file, reducing records to the Highway Accident System format. Location text fields (Place, On & At) are written to the separate LOCTEXT file.

Job U05 Creates the HSBFATAL file and CSVFATAL file from the accident data.

Job U10 Extract the fatal accidents.

1. Program THAS020 reads the HSB Fatal file and the file of new accident records, and extracts the records which match those on the HSB Fatal File. The location and jurisdiction codes from the HSB Fatal File over-ride if they differ from those on the new accident records. Jurisdiction 1 and jurisdiction 2 and 3 fatal records are written to separate files. The Total Killed field is checked on the accident records as a check for HSB Fatal File completeness.

3. The jurisdiction 1 fatal records are run through the location code checking program (THAS030) in case there are coding errors on the HSB Fatal File. Police Code and Location Type inconsistencies may also be looked for.

4. If THAS030 found any errors, an edit file is created to allow the errors to be corrected using the TSO editor. If there were location code errors, it may be safer (but not cheaper) to fix the HSB Fatal file, then re-run this job. In the case of Police Code and Location Type errors, however, the TSO edit facility must be used.

5. It is important that if any Location Code changes are made using the TSO edit facility, that the HSB Fatal file be similarly modified to ensure consistency.

Job U15 Effect the Fatal edit, and recheck

1. This job should be re-run until there are no fatal records which have failed the location code and consistency checks.

2. See the description of Job U25, which performs the same function for the non-fatal records, for more details

Job U20 - Check non-fatals and split by Location Code

1. The Location Code checking program THAS030 is run to separate the non-fatal accident file into a file of records with BLANK Location Codes (80% of all submitted MV104 accident forms), a file with INVALID Location Codes (up to 20,000 records), and a file with VALID Location Codes. This program also:

   - checks that the police code field is consistent with the Location Code. (This check may be turned off.)
   - checks that the Location Type is consistent with the Landmark Type if the Location Code is valid. (This check may be turned off.)
   - parses invalid Location Codes and checks LKI files to see if valid information can be found.

2. The file of records with invalid location codes is further divided by removing all records which have a Surrey Police code and a Jurisdiction Code of 2 or 3.

3. The file of accident records to edit is read, and an edit file is created, containing the fields required for editing and a few information fields. The fields in error are flagged. The file is sorted in Police Code order. The following fields may be edited:
   Police Code, Location Code and Location Type

4. The Edit file contains ONE abbreviated accident record per line. Only page 1 records will be on the Edit file. Changes made will be copied automatically onto records with subsequent page numbers.

6. The Location Code as originally keypunched, without spaces inserted between the sub-fields, is displayed so that incorrect sub-field boundaries can more easily be spotted. This is in addition to the editable version with separated sub-fields.

Job U25 - Put the Manual Edit into effect, and recheck the results.

1. Program THAS054 copies the fields from the edited Edit file back onto the file of unabbreviated records, and splits off the records which have been marked uncorrectable. Records marked with a V are split off to bypass the following location code check and go directly into the 'valid' file. Records marked with X, 1, 2 or 3 are split off to bypass the location code check, and go directly into the 'invalid' file.

2. The location codes are checked again. Valid ones are added to the file with valid codes ,and the others go into a file which must be edited again.

3. A new (hopefully smaller) edit file is created ready for more manual editing. If there were no errors, the Edit file will be empty.

4. This job should be re-run until there are no bad records left.

Job U30 - Merge and divide by Jurisdiction

1. The file with invalid Location Codes and the file with blank Location Codes are combined, then separated into jurisdiction 1 and jurisdiction 2 or 3 files.

2. The Jurisdiction Codes on the file of records with valid Location Codes are all set to 1.

3. The jurisdiction 1 fatal accident records are added to the file of non-fatal jurisdiction 1 records (i.e., added to the file with valid Location Codes).

4. The Causal Factor Weights are calculated for each accident, and added to the jurisdiction 1 accident records.

5. The jurisdiction 2 and 3 fatal accident records are added to the file of non-fatal jurisdiction 2 and 3 records.

6. The jurisdiction 1 fatal records with invalid locations are sorted by date, ready for merging in the next job.

6. Thus there are four final files of the update:

   Fatal Jurisdiction 1 with invalid location codes
   Jurisdiction 1 with valid location codes
   Jurisdiction 1 with invalid location codes
   Jurisdiction 2 and 3

   The file with valid location codes is sorted by Location Code. The other two are sorted by Accident Date.

7. The four final files are read by program THAS075, and accident counts are listed.

Job U38 - Update LOCTEXT Master Files

1. The .JUR1.VALLOC file (accidents with valid location codes) is sorted by Case and Date so that it can be coordinated with the .LOCTEXT file.
2. Program 085 coordinates the two files, extracting non-blank LOCTEXT records for the new accident records, creating file .LOCTEXT.VALLOC
3. .LOCTEXT.VALLOC is merged with the current THASP.LOCTEXT.MASTER file, creating an new enlarged LOCTEXT.MASTER
4. The VSAM version of the LOCTEXT file is replaced from the new LOCTEXT.MASTER file.

Job U40 - Update Current Master Files

1. The four final update files are merged with their corresponding Current Files.

2. Program THAS075 reads the new Current files, checks for problems, and lists accident counts.

3. Backup copies of the new Current files are created.

4. Archive copies of the key files of the Update are created.

Job U45 - Remove old data from Current Files

1. For each of the 3 Current files, there is an 'Archive' file, which contains all the data older than on the Current files.

2. This job takes all records from before a specified year from the Current Files, and moves them to the Archive files.

3. Backup copies of the new Current and Archive master files are created.

Job U90 - Clean up disk files

1. All the disk files created during the Update procedure (that is, all files with THASP.UPyyyymm as their first two index levels), except the Job Log and HSBFATAL files, are deleted.

7.15 Program Descriptions

7.15.1 Summary

Main Routines

```
THAS005 - extracts fatal accidents to create the HSB/CSVFATAL files.
THAS010 - copies data from the MVB file (old, pre-2001 version).
THAS011 - copies data from the MVB file (old, pre-2004 version).
THAS012 - copies data from the MVB file.
THAS020 - extracts fatal accident records.
THAS030 - checks location codes.
THAS032 - creates the Segment Table from the LKI files.
THAS040 - removes Surrey jurisdiction 2 and 3 records.
THAS050 - creates the Edit file.
THAS054 - effects the edit.
THAS060 - splits data by jurisdiction codes.
THAS061 - sets jurisdiction codes to 1.
THAS071 - creates the completion Job Log record.
THAS072 - sets the step completion code to a given value.
THAS075 - counts records and accidents, and checks for errors.
THAS080 - extract key fields from an accident file (make CASELIST)
THAS081 - reports duplicate case-date in a CASELIST file.
THAS085 - Extract non-blank LocText data corresponding to an acc. file.
THAS100 - removes old records from a (master) file.
```

Subroutines

```
THASABT - aborts a program and prints a message if a serious error occurs.
THASHNL - separates the highway letter and number into separate fields.
THASMES - prints a given message, with accident information.
THASPRC - prints record and accident counts, for program summaries.
THASRJS - right justifies a string.
THASZKM - zero fills a KMMARK field
```

Notes on the following program descriptions:

1. The acronyms in the Input and Output sections are the PL/I file names, and thus the DD names to be used in JCL executing the program.

2. The program descriptions were copied into this manual from the design specifications document. Although the descriptions were updated during development, the programs may do more than the descriptions specify.

7.15.2 THAS011 - Copy data from MVB file

```
Purpose:    to create an HSB Accident File from a MVB accident file.

Input:      - MVB       - MVB Accident file

Output:     - HSB       - HSB Accident file
            - LOCTEXT   - text location information.
            - SYSPRINT  - messages and record counts
```

Description:

- Read each MVB record, copy selected fields into the 320 byte HSB ACDAT_RECORD, and write to DDNAME HSB.

- Copy the Case, Location, On and At text information to the LOCTEXT record, and write it.

- Copy the location code (11 characters) unmodified into the HSB field MVB_LOCN.

- The location code is copied into LOCN_CODE, with the highway letter separated into its own sub_field. Make sure the three character MVB Highway field is right justified before trying to decode it.

  Make the Highway number and letter separation code into a separately compiled subroutine, so that it can be used by other programs.

- If the Page Number is > 1, make sure that:

    - the page number is one greater than the page number of the previous record,
    - the police code, accident case number, location code, jurisdiction and date are the same as on the previous record.

  If any of these conditions are not true, print a warning message with the accident case number, location code and the discrepancy.

- Print a count of records read, and records written.

- Print a count of the number of 1, 2, 3, ... page accidents, and a count of the total number of accidents.

7.15.3 THAS020 - Extract Fatal Accident Records

Purpose: to split the HSB Accident File into non-fatal, fatal jurisdiction 1, and fatal jurisdiction 2 and 3 files.

Input:
- INACC - HSB Accident file
- HSBFAT - The Highway Safety Branch's manually entered Fatal Accident File.

Output:
- NONFAT - Non-fatal Accident file
- FATJUR1 - Fatal jurisdiction 1 accident file
- FATJUR2 - Fatal jurisdiction 2 and 3 accident file
- SYSPRINT - messages and record counts

Description:

- Both the input files are sorted in Accident Case and Page Number order.

- Read the INACC and HSBFAT files, and match on Accident Case and Page Number. The highway field on the HSBFAT file will have to be separated into highway number and letter fields, using the routine written for THAS010, before comparisons can be made.

- If the INACC record has no HSBFAT match:

    - Report the record if the Total Killed field > 0
    - Write the record to file NONFAT

- If the INACC record has an HSBFAT match:

    - Compare the Location and Jurisdiction codes on the two records. If they differ replace with the codes from the HSBFAT file, then report, giving the Accident Case Number, INACC data, and the HSBFAT data used.

    - If the Total Killed field on the INACC file = 0, report.

    - Write to the FATJUR1 or the FATJUR2 files, depending upon the Jurisdiction Code.

- If a case number on the HSBFAT file does not have a match on the INACC file, report.

- Print record counts in and out, and counts of each type of warning message produced (eg 13 HSBFAT records had no INACC matches, etc.)

7.15.4 THAS030 - Check Location Code

Purpose: to split an Accident File into files with valid, invalid, and, optionally, blank Location Codes.

Input:
- INACC - Accident file
- LMATCH - control file of location type matches
- SEGTAB - Segment Table file, created by THAS032
- TLKIHWY - LKI Highway file, read by routine TLKIC02
- LMK - LKI Landmark file
- parameter - indicating whether records with blank location codes should be written to a separate file, and whether invalid location codes should be parsed to find valid sub-fields.

Output:
- NOLOC - records with blank location code (conditional)
- INVLOC - records with invalid location code
- VALLOC - records with valid location code
- SYSPRINT - messages and record counts

- Return code - 2 if any Location Code errors were found
  - 1 if only Police or Location Type errors

Description:

- Read the LMATCH file into a table of corresponding Landmark Types and Location Types.

- Examine the EXEC parameter character string. If it contains 'BLANK' then set a flag indicating that records with blank location codes are to be written to NOLOC. If it contains 'PARSE' set a flag indicating that invalid location codes are to be parsed to try and find valid sub-fields.

- Read and process each page 1 record on the INACC file as described below. Records with page number > 1 will not be checked, but will be written to whatever file their page one was written to.

- If the $PARSE_FLAG is on, write the accident to INVLOC. This means that the location code was made valid by a previous THAS030 run, but was not marked 'Checked' with a C by the user on the edit file. The accident will keep returning to the edit file until it has been checked. (Program THAS054 turns off the $PARSE_VALID flag if the record is marked with a 'C'.)

- Check the LOCN_CODE field:

  - If the location code is blank, and the Jurisdiction = 2, write the record to either the NOLOC or INVLOC files depending upon the BLANK flag.

  - Check that the highway number and letter are on the LKI Highway File.

  - Check that the segment number is on the LKI Segment File.

  - Check that the highway-segment combination is on the LKI highway/Segment file.

  - Check that the KMMARK is valid (i.e. numeric and positive), then check that it is a valid distance in the segment.
    (0 <= KMMARK <= segment length in the LKI Segment file.)

- If the location code is not valid, set the appropriate bit in the LOCN_ERROR field. (See the description of the LOCN_ERROR field below.)

- If the segment and kmmark are valid, check the location type: (this is the 2 byte LOCN_TYPE field, called 'Accident Location' on the MV104 form.)

- If the accident location (segment and kmmark) matches a landmark location on the LKI Landmark file, then if the Landmark Type is on the LMATCH table, the Location Type on the accident record must equal the corresponding Location Type on the LMATCH table. If it does not, set the $BAD_LOC_TYP bit of the LOCN_ERROR field.

  NOTE: always round the LKI km's to 1 decimal place before comparing to an accident record KMMARK.

- If location code is not valid, and the PARSE flag is on, examine the MVB_LOCN field as described in the 'PARSE' description below. If the LOCN_CODE field is modified as a result, but is still invalid, set the $PARSE_INVAL bit of the LOCN_ERROR field. If a valid location code results, set the $PARSE_VALID bit.

- If the segment number is valid, check that the Police Code is consistent with the segment number by checking the segment/RCMP detachment file of the LKI.

  If there is no record on the LKI file with the segment/police code combination, set the $BAD_POLICE bit of the LOCN_ERROR field.

- If there are no bits set in the LOCN_ERROR field, write to the VALLOC file. Otherwise, write to the INVLOC file.

Error Bits in the LOCN ERROR field

Bit	Name	Description
1	$BADHWY	the highway number and/or letter is invalid.
2	$BADSEG	the segment number is invalid.
3	$BAD_HWY_SEG	the segment and highway are incompatible
4	$BADKM	the KMMARK is invalid, or > the segment length
5	$BAD_LOC_TYP	the location type does not match the LKI
6	$BAD_POLICE	the police code is invalid for this segment
7	$PARSE_VALID	the locn code was modified to make it valid
8	$PARSE_INVAL	the locn code was modified but still invalid

A bit will be set to 1 (TRUE) to indicate the condition described for that bit.

Member THASRACO of THASx.TEXTLIB can be included into a program after the Accident Data record definition to overlay-define the individual bit names on the LOCN_ERROR field.

Location Code Parsing

```
MVB_LOCN:    1 2 3   4 5 6 7   8 9 10 11
             highway segment   kmmark


Object:  to fill in as many of the LOCN_CODE subfields as possible.

(In the following pseudocode, column numbers refer to the MVB_LOCN field.)

If the LOCN_CODE highway number and/or letter is invalid, then:
      If cols 1-3 are blank then:
            If the LOCN_CODE segment and kmmark fields are valid, then:
                  Look up the highway information on the LKI Highway/Segment
                  file, and insert into the LOCN_CODE.
            Endif
      Else:
            If the first four consecutive nonblank characters are a valid
            segment then:
                  Insert them into the LOCN_CODE segment field.
                  Find the next set of consecutive non-blank characters.
                  If the length of that field is 4 or 5, then:

                        Decode a km distance from the field, with one decimal
                        place if the length is 4, and 2 decimal places if the
                        length is 5.

                        If a valid KMMARK for the segment is obtained then:
                              Insert into KMMARK
                              Look up the highway on the LKI Highway/Segment
                              file and insert into the Highway fields.
                        Endif
                  Endif
            Endif
      Endif

Else: (highway is valid)
      If the segment is valid, but the KMMARK is too large then:
            If the characters in cols 8-11 are not right justified then:
                  Right justify cols 8-11, and re-decode the kmmark
            Else:
                  If all 4 columns are occupied, read with 2 decimal places
                  instead of one.
            Endif
            If it is now valid, insert into LOCN_CODE KMMARK.
      Endif
Endif
```

7.15.5 THAS032 - Create Segment Table

```
Purpose:    to combine the information from several LKI files into one file
            containing the segment-keyed LKI information needed by program
            THAS030.

Input:      - SEG       - LKI Segment file
            - TLKIIHS   - LKI Highway/Segment intersection file
                          (looked after in LKI routine TLKIC06)
            - TLKIHWY   - LKI Highway file.
                          (looked after in LKI routine TLKIC02)

Output:     - SEGTAB    - Segment Table File.
            - SYSPRINT  - messages and record counts


Description:

- for each record on the Segment File, create a Segment Table record
  containing the segment number, highway number, and segment length.

- if there are more than one Highways listed for a Segment, use the first
  Highway.

- the Highway file is used only for verifying the Highway taken from the
  Highway/Segment file.
```

7.15.6 THAS040 - Remove Surrey Jurisdiction 2's

Note: THAS040 is no longer used. Surrey location codes are now translated into H.A.S. location codes. See the description of the THASP.REXX.EXEC(FIX<xxxx>) files in the User's Manual.

Purpose: to split into a separate file all the Surrey accident records which have a Jurisdiction code of 2 or 3.

Input:
- INACC - Input accident file

Output:
- SURREY2 - File of Surrey jurisdiction 2 and 3 records
- TOEDIT - All other records
- SYSPRINT - messages and record counts

Description:

- Read INCACC, and if the Police Code = 726 and the Jurisdiction Code = 2 or 3, then write to SURREY2, else write to TOEDIT.

- Also, count the number of Police Code 726 records which have a Jurisdiction code = 1.

- Print the number of records read, and written to each file. Print the number of Surrey records with jurisdiction = 1.

Notes:

- The Surrey police use a municipal location code instead of the provincial one. Since at present there is no way of converting from the municipal to the provincial location code, there is no point in putting them on the error file to be edited. Jurisdiction 1 records are sent to be edited because if the policeman is aware that a road is provincial jurisdiction, there is a chance the provincial location will be used.

- This program could be expanded any time a set of records can be defined which have invalid location codes but do not need to be manually edited or checked.

7.15.7 THAS050 - Create Edit Records

Purpose: to create PDS members containing only the records and fields required for the manual edit process.

Note: this program combines the functions of THAS050 and THAS052 of the original design.

Input:
- INACC - Input accident file

Output:
- E1 - First member
- E2 - Second member
- ...E9 - subsequent members (maximum of 9)
- SYSPRINT - messages and record counts

Description:

- Read INACC record, and if the page number is 1, create an Edit record as described below, then write it to file En.

- Note: the 'police subdivision' is the leftmost 2 digits of the police code.

- Write to DDname En until there are more than 10,000 records written to En, AND the police subdivision changes.

- Then close En, increment n, open En, and start writing to En.

Edit Record Creation

- The edit record Action field (col 1) should be set blank.

- Insert a vertical line either side of the PARSE field.

- Insert an underscore before and after the SEGMENT field of the Location Code.

- Insert error flags, depending upon the flags in the LOCN_ERROR field, as follows:

```
      if $BADHWY       put * in HWY_ERR
         $BADSEG            *    SEG_ERR
         $BAD_HWY_SEG       #    HWY_ERR
         $BAD_KM            *    KM_ERR
         $BAD_LOC_TYP       *    TYP_ERR
         $BAD_POLICE        *    POLICE_ERR
         $PARSE_VALID       V    PARSE_FLAG
         $PARSE_INVAL       X    PARSE_FLAG
```

- Count and report:
  - records read in
  - records for each police code
  - records for each police subdivision
  - records written to each member.

7.15.8 THAS054 - Effect the Edit

```
Purpose:    to put the changes made on the edit file into effect on the
            TOEDIT accident file.

Input:      - EDIN      - File of edited edit records
                          (The PDS members concatenated)

            - TOEDIT    - Accident file to edit

Output:     - EDITED    - Edited accident file
            - OVRIDE    - Records manually declared valid - not to be
                          rechecked
            - GIVEUP    - Invalid records - not to be re-checked
            - SYSPRINT  - messages and record counts

Description:

- The two input files will be in Police Code order, and every page 1 on file
  TOEDIT should have a record on file EDIN with matching Accident Case, Page
  number and Accident Date.  If a mismatch is encountered, the program should
  abort, after printing the record numbers and identification fields of the
  last records read on each file.

- Page 1 TOEDIT records and any continuation records (with page number > 1)
  are processed as described below, using the corresponding EDIN record.

- Print record and accident counts in and out.

Record Processing:

SELECT CASE on the Action field of EDIN (col 1):

      =blank:

         -  remove leading zeros and right justify the Highway number field
            of the EDIN record.
         -  copy the Police Code, Location Code and the Location Type fields
            from the EDIN to the TOEDIT record.
         -  make sure that the KMMARK field is zero filled.
         -  write the TOEDIT record to the EDITED file.

      =C:

         -  set the $PARSE_VALID flag of the accident record to '0'B.
         -  then process exactly as for 'blank'.

      =V:

         -  same as 'blank', but write to file OVRIDE

      =1, 2, or 3:

         -  put 1, 2, or 3 into the Jurisdiction Code of the TOEDIT record.
         -  write to file GIVEUP

      =X:

         -  write the TOEDIT record unchanged to file GIVEUP.

  END SELECT CASE.
```

7.15.9 THAS060 - Split Data by Jurisdiction Code

```
Purpose:    to divide an accident file into two files, depending upon the
            Jurisdiction Code field.

Input:      - INACC     - Input accident file

Output:     - JUR1      - Jurisdiction one records
            - JUR2      - Jurisdiction two and Jurisdiction three records
            - SYSPRINT  - messages and record counts

Description:

- Read from INACC, and write to JUR1 or JUR2 depending upon the jurisdiction
  code.

- If the jurisdiction code is neither 1 nor 2 nor 3, write a warning message,
  count, and write to JUR2.
```

7.15.10 THAS061 - Set the Jurisdiction Code to 1

```
Purpose:    to set all the jurisdiction codes on an accident file to 1.

Input:      - INACC     - Input accident file

Output:     - JUR1      - Jurisdiction one records
            - SYSPRINT  - messages and record counts

Description:

- Read from INACC, count the records with and without a jurisdiction code
  which is already 1, set to 1, and write to JUR1.

- Print record counts.
```

7.15.11 THAS100 - Remove Oldest Records from a Master File

```
Purpose:    to remove all the accidents dated before a given year from a
            Current Accident File and put them on the corresponding Archive
            Accident File.

Input:      - INACC     - Input accident file
            - parameter - Oldest Year to keep (2 digits, e.g. '84')

Output:     - OLD       - Archive accident file
            - CUR       - Current accident file

            - SYSPRINT  - messages, record and accident counts

Description:

- Get and check the 2 digit year from the program EXEC parameter.

- Read a record from the INACC file, and if the date is prior to the Oldest
  Year, write it to the OLD file.  Otherwise write it to the CUR file.

- Print the Oldest Year, and the oldest and newest dates of accidents written
  to the CUR file.

- Print all record and accident counts.
```

7.15.12 THAS140 - Calculate Causal Factors

Purpose: Determine causal factors (or weights) for each accident, to apportion the blame for the accident out to each of the Driver, Vehicle and Road, such that the sum of the three weights is 1.0.

Input:
- INACC - input accident file
- LABELD - file of labelled observations
- CFRANK - table for converting a contributing factor variable from the accident record into a ranked variable in the observation.

Output:
- OUTACC - output accident file, with vehicle and road weights inserted in all page 1 records.

Terminology

Observation
- a set of 20 numeric variables derived from an accident record.

Label
- the three causal factor weights Wd, Wv, and Wr (in that order).

Labelled Observation
- an observation with a Label appended.

Unlabelled Observation
- an observation without a label
- the data derived from the accident we want to calculate weights for.

K - the number of 'nearest' labelled observations to be found and used to determine the weights for the unlabelled observation.

Algorithm Summary

A fuzzy pattern recognition algorithm is used. The program is based upon a C++ program written by Tarek Sayed.

A file of 1000 Labelled Observations was created by Tarek Sayed and Walid Abdelwahab of the Highway Safety Branch. The 20 variables were defined from the accident data, then the weights were assigned manually.

For the candidate accident, (unlabelled observation), the 20 variables are defined, then the observation is compared to all of the Labelled Observations. The K Labelled Observations which match the Unlabelled Observation most closely are found, then the weights of those observations are used to determine the weights for the accident.

References

Tarek Sayed was expecting his paper to be published some time in 1994 or 1995:

Journal of Transportation Engineering
(American Society of Civil Engineers)
"Application of Fuzzy Pattern Recognition Techniques to the Identification of Accident Prone Locations."

Storage of Weights in the Accident Record

To conserve space, the Vehicle and Road weights (Wv and Wr) are stored in the accident record as 2-digit percentages. The Driver weight, (Wd) if needed can be calculated as 100 - (Wv + Wr). In the unlikely event that a percentage is greater than 99.5, it is to 99. The Driver weight is the one left out, because it is the one which is most likely to be 100.

The Vehicle Weight is put in bytes 34 & 35 of the accident record, and the Road Weight is put in bytes 36 & 37. (These are the first 4 bytes of the obsolete Accident Case Continuation field.)

Program Outline

- read in K, and NMAX (the number of labelled observations to use.)
- read in the labelled observations (LABELD).
- read in the contributing factor rank table (CFRANK)

- normalize the labelled observations (i.e. reduce all variables to a real number between 0 and 1.) The minimum value of a variable becomes 0, the maximum becomes 1., and all others are placed in between. The minima, maxima, and ranges for each variable are kept for use in normalizing the unlabelled data later.

- FOR each accident: (page 1's only, just copy others)

    - convert the accident to an 'observation'. This is done according to the table included below.
    - normalize the observation, using the min, max and range values kept from the labelled observations.
    - calculate the 'distance' from the unlabelled observation to each of the labelled observations, keeping track of where the K 'nearest' labelled observations are. The distance between two normalized observations a and b, each composed of p variables, is calculated as follows:

$$d = \sqrt{\sum_{i=1}^{p} (a_i - b_i)^2}$$

- determine the weights for the unlabelled observation from the weights of the K nearest labelled observations. For each of the three weights (driver,vehicle and road), this is done as follows:

$$WT = \frac{\sum_{i=1}^{k} \frac{w_i}{d_i^2}}{\sum_{i=1}^{k} \frac{1}{d_i^2}}$$

where:
- WT is the weight being calculated,
- W_i is the weight of one labelled observation
- d_i is the distance from a labelled observation to the unlabelled observation.

- normalize the three calculated weights, to ensure they add to 1.

- convert the vehicle and road weights to integer percentages, and put them into the accident record.

- write out the accident record.

ENDFOR

Converting Accident Variables to an Observation

Table 1. Accident variables and their levels.

Variable	Levels
1. Degree of curvature ROADCURV	1. Straight 2. Single Curve 3. Sharp Curve
2. Weather condition WEATHER	1. Clear/Cloudy 2. Raining 3. Ice/snow 4. Smog/fog
3. Land use LANDUSE	1. Undeveloped/agriculture Area 2. Rural residential area 3. Urban residential 4. Central Business District
4. Road grade ROADGRAD	1. Flat 2. Some grade 3. Steep grade 4. Hillcrest/sag
5. Surface condition ROADSURF	1. Dry 2. Wet 3. Ice/snow
6. Speed limit SPEEDLIM	1. <=60 kph 2. 70-80 kph 3. 90-110 kph
7. Traffic control device TRAFCNTL	1. Exist 2. None
8. Lighting conditions LIGHTING	1. Day light 2. Dark/Full illumination 3. Dark/Some illumination 4. Dark/No illumination
9. Accident location LOCN_TYPE	1. At intersection 2. Not at intersection
10. Use of a restraint device VICTIM_TABLE(1).SAFQUIP	1. Device used 2. Vehicle equipped but device not used 3. Vehicle not equipped
11. Accident time	1. Non-rush hour 2. Rush hour: 7-9, 12-13, 16-18
12. Vehicle type	1. Passenger cars only 2. At least one van or pickup 3. At least one truck or bus
13. Accident type	1. Single vehicle-fixed object 2. Single vehicle-other 3. Multiple vehicle-head on 4. Multiple vehicle-side/angle 5. Multiple vehicle-rear-end 6. Pedestrian 7. Animal
14. Severity	1. Property damage only 2. Injury 3. Fatal

In addition to these 14 variables there are another 6 variables that describe accident contributing factors (3 factors for each vehicle). The H.A.S. PL/I names for these variables are:

CONTRB11
CONTRB12
CONTRB12
CONTRB21
CONTRB22
CONTRB23

The numeric codes are re-assigned for the observation, as follows:

1. Alcohol involvement	20. Cutting in	63. headlights defective
2. Driver inexperience	21. Driving on wrong side of road	64. Turn Signal defective
3. Drugs	22. Improper turning	65. Oversize vehicle
4. Extreme fatigue	23. Failing to yield right of way	66. Steering failure
5. Fell asleep	24. Ignoring traffic control device	67. Tires - failure/ inadequate
6. Illness	25. Pedestrian error confusion	68. Two hitch failure
7. Sudden loss of consciousness	40. Obstruction on Road	69. Driverless vehicle
8. Pre-existing physical disability	41. pavement surface defective	70. Windshield defective
9. Prescribed medication	42. Road maintenance/construction	71. Engine failure
10. Attempted at suicide	43. Sign obstruction	72. Suspension defect
11. Driving without due care	44. Insufficient traffic control	73. restraint system
12. Failing to signal	45. Road/intersection design	74. Insecure load
13. Ignoring officer	46. Roadside/hazard	75. Dangerous good
14. Previous traffic accident	47. Wild animal	76. Vehicle modification
15. Following too closely	48. Weather	77. Glare artificial
16. Improper passing	49. Visibility impaired	78. Glare sunlight
17. Unsafe speed	60. Accelerator defective	79. Domestic animal
18. Avoiding vehicle/pedestrian/cycle	61. Brakes defective	99. Others
19. Backing unsafely	62. Brake lights out	

8 Data Retrieval Sub-System

8.1 Design

8.1.1 Introduction

See the **Data Retrieval Concepts** section of the User's Manual for an introduction to the concepts and capabilities of the system.

8.1.2 Data Selection

Selecting data from the PDS Master is always done as a separate step at the beginning of the job. The output from the step is a temporary (unless the user wants it cataloged) Subset Pair. This Subset is read by all processes which are to get their input from the 'Initial Data Source'.

8.1.3 JCL and the ISPF Dialog

See the **REXX/ISPF Dialog** section for a description of the ISPF Dialog.

The JCL is coded as ISPF File Tailoring Skeletons in library THASP.ISPF.ISPSLIB. There is one skeleton for the job statements of a Data Retrieval job (member THASDRET), and there is a skeleton for each Data Retrieval Process.

When the user selects a process, and specifies process parameters, the ISPF Dialog uses the File Tailoring facility to take the JCL skeleton for that process, substitute parameters for variable names, and append the JCL to the <userid>.JOBJCL file. When the user has finished defining the Data Retrieval job, the JOBJCL file can be submitted as a batch job.

Standard DD names are used in the JCL for Selected and Excluded accident data subsets so they can be referenced in later processes. Thus if the excluded data from step B is required, the dataset is specified as a referback to STEPB.XDATA.

See the REXX program and JCL skeleton THASP210 for an example of this procedure.

8.1.4 Temporary File Names

Temporary subset pairs which are used to pass data between steps are named as follows:

```
      &&ptDATA and &&ptDESC

where 'p' is the process letter (A,B,C ...) and

      't' is the type: either S for included (Selected) records, or
                              X for eXcluded records.
```

Thus the output from the DATA SELECTION program is files &&ASDATA and &&ASDESC. If the Accident-Prone Section program were run next, the records included in accident-prone sections would go to file &&BSDATA, described by file &&BSDESC, and all the other records which were in the input file would go to file &&BXDATA, described by &&BXDESC.

8.1.5 Step Names

The step for process 'D' is named STEPD. If a process requires more than one job step, the step which produces output subsets (if any) is the one which should be called STEPD, for referback purposes. Preceding or following step names do not matter in the context of the whole job.

The step name of the step which produces output subsets is coded as '&STEP.' in the skeleton JCL. The current step letter is maintained in REXX routine THASPSEL, and passed to routine THASJCLS, which defines variable STEP before invoking the file tailoring service.

8.1.6 DD Statements

Each process needs an input subset pair, and 0, 1 or 2 output subset pair DD statements. The input DD names must be INDATA and INDESC. The output DD names must be SDATA and SDESC for Selected (or included) records, and XDATA and XDESC for eXcluded records.

If a process can produce output subsets, but they are not needed in future steps, then time is saved by telling the program not to write the records out (with flags in the SYSIN control parameters). The corresponding DD statements are then dummied out (by setting DSN=NULLFILE).

The input Data Set Names are refer-backs, constructed from the input file information specified by the user. If the input is to be the Exclusion subset from process B, for example, the input DD statements are as follows:

```
//INDATA DD DSN=*.STEPB.XDATA,DISP=(OLD,PASS)
//INDESC DD DSN=*.STEPB.XDESC,DISP=(OLD,PASS)
```

8.1.7 Long Search Paths

The description file, including the full search path, is printed out at the beginning of each report. It is also printed when the data is selected, by THAS200. The search path may be quite long, if a large part of the provincial highway system is chosen. To save paper, therefore, THAS200 prints only the first 10 elements of the search path. The full search path is still included in the description file, and thus the full search path will be printed at the beginning of each Data Retrieval program's output.

8.1.8 To Add a Process

a) Make sure the program reads from DD names INDATA and INDESC.

b) The program should print the contents of INDESC on any reports it produces.

c) If the program produces one or two output subsets, the DD names described above must be used. Make sure Subset Pairs are produced. See the definition of Subset Pairs in the User's Manual. See PLI subroutine THASDES.

d) Write JCL for the step. See THASP.ISPF.ISPSLIB(THASP210) for an example of how to code &xxx. parameters for substitution.

e) Write a REXX program in THASP.ISPF.ISPCLIB for the new process:
   - See THASP240 for a simple example, and THASP210 for a more complex one.
   - Create panels to get control parameters from the user. (See THASP.ISPF.ISPPLIB(THASP210) for an example of a panel.)
   - Call EXEC THASPECS if the process creates subsets.
   - Prepare parameters for and call THASJCLS, which supervises the JCL Skeleton file tailoring to add JCL for the process to the JOBJCL file.

f) Add a line for the process in the Process Specification Menu in panel THASPSEL, and add required code to the REXX program THASPSEL.

8.2 Program Descriptions

8.2.1 Summary

Main Routines

THAS200 - Extracts data from the PDS Master Files.
THAS203 - Select accidents from a subset by Data Fields.
THAS205 - Select accidents by counter-measure accident type.
THAS210 - Accident-Prone Location main program.
THAS220 - Accident-Prone Section main program.
THAS225 - Specified Sections Analysis.
THAS230 - Histogram main program.
THAS232 - Fatal/Injury/PDO Accident Counts.
THAS240 - Details Report main program.
THAS250 - Summary Report main program (version 1).
THAS251 - Summary Report main program (version 2).
THAS260 - Rate Table main program.
THAS270 - Average Accident Type Ratios.
THAS710 - Create Victim File.

Subroutines

See Appendix A.

Notes on the following program descriptions:

1. The acronyms in the Input and Output sections are the PL/I file names, and thus the DD names to be used in JCL executing the program.

2. The program descriptions were copied into this manual from the design specifications documents. Although the descriptions were updated during development, the programs may do more than the descriptions specify.

8.2.2 Program THAS200 - Data Selection

```
Purpose:  to extract data from the PDS master file.

Input:      - SYSIN     - selection criteria (from the menu)
            - SEGPDS    - the Segment PDS of the PDS Master
            - NODPDS    - the Node PDS of the PDS Master
            - SHNFIL    - the Segment-Highway-Node file
            - SEGCLAS   - the sorted Segment Classifications file
            - SEGDIST   - the sorted LKI Segment-Districts file
            - TLKIDST   - the LKI Districts file
            - LMKFIL    - the LKI Landmark file
            - RGFFIL    - the (RFI) range features file
            - FLDNAM    - HAS field data - THASP.TABLE(FLDNAMES)
            - SEGFILE   - the LKI SEGMENT file (for routine SIH)

Output:     - SDATA     - selected accident records
            - SDESC     - description file for SDATA
            - SYSPRINT  - selection criteria and record counts

Description:

- create a list of From-To specifications that is the intersection of the
  districts, highway classifications, and selected From-To specifications
  read from SYSIN.

- create an ordered list of segments and nodes required, using the highway
  network subset specifications calculated above, and the highway-segment-
  node information read from SHNFIL.

- if the Node Selection mode is '1', make sure that each node occurs only
  once on the list.  If it is 'M', nodes should repeat as required to
  complete the path along each highway.  If it is '0', skip all nodes.

- using the assembler PDS access utility, (see PLI program THASPDS) read the
  listed PDS members in order, and select accident records according to the
  other selection criteria specified by the user.

- if data codes were specified, check that each accident meets the code
  requirements.

- if landmark codes were specified, check the location code against the
  landmark file, and keep only those accidents which occur at one of the
  specified landmarks.

- if the Range Features flag is set, check the location code against the
  range features file, and keep only those accidents which are spanned by a
  range feature.  (The Range Features file must be reduced to only the
  desired range features prior to running THAS200.)

- look up and the highway classification for each selected accident, and
  insert it into the accident records.

- write the selected accident records to SDATA.

- write program ID, date, and selection criteria to the SDESC file.

- write the search path (list of segments and nodes) to the SDESC file.  (See
  the definition of the SDESC file for details.)
```

Algorithm used to generate intersections of FROM-TO lists

Construct a From-To list containing all segments and sections of segments in the selected districts. Construct another From-To list containing all segments and sections of segments with any of the selected highway classifications. The third From-To list is the one specified directly by the user.

Given are the three From-To sets FTA(1..m), FTB(1..n), FTC(1..p).
Add the HSNTAB sequence numbers to each member of each set. These are the indexes in the HSNTAB of each segment number (From and To). Now each From-To specification has the following 6 fields:

FSEG	TSEG
FKM	TKM
F#	T# (the sequence numbers)

```
For i = 1 to m:
   For j = 1 to n:
    ¦ Find the intersection of FTA(i) with FTB(j).  Call it FT1.
    ¦ If FT1 is non-null:
    ¦  ¦ For k = 1 to p:
    ¦  ¦  ¦ Find the intersection of FT1 with FTC(k).  Call it FT.
    ¦  ¦  ¦ If FT is non-null:
    ¦  ¦  ¦    Add it to the final From-To set.
```

To get the intersection of two individual From-To specs (FT1 and FT2) and call it FT:

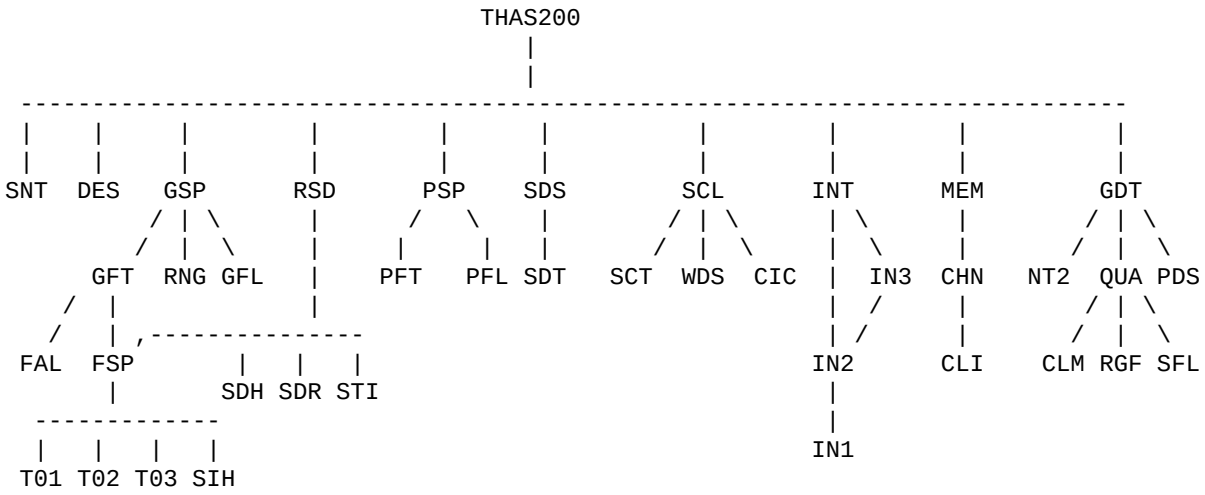
FT.F# = max(FT1.F#, FT2.F#).
FT.FSEG is FT1.FSEG or FT2.FSEG, depending on which F# was the max.
FT.FKM is similarly FT1.FKM or FT2.FKM. If FT1.F# = FT2.F#, then FT.FKM is max(FT1.FKM, FT2.FKM).

FT.T# = min(FT1.T#, FT2.T#).
FT.TSEG is FT1.TSEG or FT2.TSEG, depending.
FT.TKM is FT1.TKM or FT2.TKM. If FT1.T# = FT2.T#, then FT.TKM is min(FT1.TKM, FT2.TKM).

To determine whether the intersection is null:

If FT.F# > FT.T#, the intersection is null.
If FT.F# = FT.T# and FT.FKM > FT.TKM, the intersection is null.
Otherwise the intersection is not null.

THAS200 Structure Chart



Utility: HNL, T03, ABT

THAS200's SYSIN data is generated and formatted by the REXX programs, which place it as a single table in the skeleton JCL. Thus, it cannot be found in the skeleton libraries; however, when JCL generation is complete, the SYSIN data looks like this:

THAS200 - Sample SYSIN File

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7
//SYSIN    DD *
900101 900331   ; Date range
05 08           ; Month range
09 18           ; Hour range
'U CE '         ; Highway Classification Set
1               ; Node Selection 0, 1 or M
YES             ; Include Obsolete Locations
GUARDRL         ; Range Feature selection SAS program
'FROM-TO:'
'97A' '    ' '     ' '   ' '    ' '     ' '    '
'3  ' '    ' '     ' '6  ' '    ' '     ' 'NO  '
'1  ' '0925' '     ' '   ' '0935' '     ' '    '
'   ' '0364' '     ' '   ' '    ' '     ' '    '
'END' '    ' '     ' '   ' '    ' '     ' '    '
'SECDEF:' 'SDTEST' 'Y'
'DISTRICTS:'
' 4'
' 5'
' 6'
' 7'
'EN'
'LANDMARKS:'
'A1'
'A2'
'A3'
'EN'
'DATA CODES:'
'CONTRB11' '#' 'CODES'         ; field name, group code, codes following
'10'
'13'
'END'
'DIAGRAM' '#' 'CODES'
'01'
'02'
'END'
'END DATA CODES'
AND                            ; data field combining logic
1                              ; check page 1 only ( or A for All pages)
/*
```

Note: For the start and end months and the start and end hours, it is possible to specify start > end. For example, it is possible to specify months 11 through 2 (November through February), or hours 18 through 6 (6:00 pm through 6:00 am, i.e. night-time hours).

8.2.3 Specifications for PDS access utility subroutines

Assembler routine THASRDS allows members of a PDS to be read at random from PL/I. The DDNAME of the PDS (SEGPDS) is hard coded in the assembler code, so to read a second PDS, a second assembler routine was created by changing the following in the code:

```
'THASRDS' to 'THASRDN'
'SEGPDS'  to 'NODPDS'
'DUMRECS' to 'DUMRECN'
```

Using these two routines, both the segment and node PDS's of the PDS Master can be read at once.

The assembler routines read data a block at a time.

8.2.3.1 PLI Routine THASPDS

To make PDS reading from a PL/I program easier, PL/I subroutine THASPDS was written. THASPDS handles calls to both assembler routines, and does the de-blocking of the data. THASPDS is designed so that it can easily be modified to handle more, or different assembler routines and DD names. It contains a list of valid DD names, and a corresponding list of assembler routine entry names.

```
THASPDS: PROCEDURE(COMMAND,DDNAME,MEMBER,REC,STAT) REORDER;

   /************************ PROLOGUE ********************************/
   /*                                                                */
   /*  PROGRAM: THASPDS                                              */
   /*  PURPOSE: To perform assembler subroutine calls and            */
   /*           de-blocking for OPEN, FIND, READ and CLOSE functions */
   /*           on the segment and node Partitioned Datasets.        */
   /*                                                                */
   /******************************************************************/

   /*------------------ Parameters ----------------------------------*/

   DCL COMMAND  CHAR(4);        /* OPEN, FIND, READ or CLOS          */
   DCL DDNAME   CHAR(8);        /* SEGPDS or NODPDS                  */
   DCL MEMBER   CHAR(8);        /* a PDS member name                 */

   DCL REC      CHAR(256);      /* record returned from READ         */
   DCL STAT     FIXED BIN(31);  /* -1  end of file on a read         */
                                /*  4  member not in the DDNAME PDS  */
                                /*  5  unsupported DDNAME            */
                                /* 98  attempt to READ before FIND   */
                                /* 99  invalid command               */

   /*---- Assembler routines which do the nitty gritty work ---------*/

   DCL (THASRDS,        /* for DD name SEGPDS */
        THASRDN         /* for DD name NODPDS */
       )EXTERNAL ENTRY OPTIONS(ASSEMBLER);

   DCL PDS_ACCESS(2)  ENTRY VARIABLE OPTIONS(ASSEMBLER)
                      INIT(THASRDS,THASRDN);

   DCL VALID_DDNAMES(2) CHAR(8)     STATIC INIT('SEGPDS','NODPDS');
   DCL MAXDD            FIXED BIN   STATIC INIT(2);
```

8.2.3.2 Assembler Routine THASRDS

This module allows you to "point" to any member in a PDS and then read sequentially (in blocks) until end of member. Note that it is up to the calling routine to unblock the records - this can be done simply by overlay-defining a set of records on top of the block. This routine can only access a file of record length 256 and blocksize of 2560.

Declare this routine as EXTERNAL ENTRY with OPTIONS of ASSEMBLER.

```
e.g.  DCL THASRDS    EXTERNAL ENTRY OPTIONS(ASSEMBLER) ;
```

Parameter 1 - Command

This is the primary command passed to the assembler routine. It should be a character string of length 4 (literals are allowed).

'OPEN' - Opens the Partitioned Dataset for reading. Open once per dataset, not once per member.
'FIND' - Positions the file to a new member, ready for subsequent reads.
'READ' - Reads the next Block of records from the current member.
'CLOS' - Closes the Partitioned dataset. Issue once after you have finished all reading.

Parameter 2 - Member Name

This is the name of the member you wish to start reading. This value is only applicable with a FIND Command. This parameter should be declared as CHAR(8).

Parameter 3 - Return Code

This value is the status of the issued command. It should be declared as FIXED BINARY(31). It will have the following exceptional values:

Command	Return Code	Meaning
FIND	4	Member not found.
READ	4	End of Member
????	99	Not a valid command

It should be 0 in all other normal circumstances.

Parameter 4 - Record Count

This parameter will tell you the number of records to be found in the block. In most cases it's the same as BLKSIZE/LRECL (i.e., 10), but it is possible for the last block to be shorter. This parameter should be declared as FIXED BINARY(31).

Parameter 5 - Block

This parameter will contain the block of records you issued a READ for. It should be declared as CHAR(2560).

JCL for THASRDS:

Within your JCL declare your file with a DDNAME of SEGPDS:

```
    e.g.  //SEGPDS  DD  DSN=THASD.SEGPDS,DISP=SHR
```

Note that you have to use a different DDNAME with the THASDIR module.

Example:

This example uses both THASDIR and THASRDS. It copies all records from all members in a PDS and writes them out to a straight sequential file.

```
  DCL THASDIR           EXTERNAL ENTRY;
  DCL THASRDS           EXTERNAL ENTRY OPTIONS(ASSEMBLER) ;

  DCL SEGPDD            FILE RECORD SEQUENTIAL INPUT ;
  DCL OUT               FILE RECORD SEQUENTIAL OUTPUT ;

  DCL MEMBERS(450)      CHAR(8) ;
  DCL N                 FIXED BINARY(31) INIT(450) /* Array Bound */
  DCL #M                FIXED BINARY(31) ;  /* No. of Members in Library */

  DCL RC                FIXED BINARY(31) ;  /* Return Code from Assembler */
  DCL #REC              FIXED BINARY(31) ;  /* Record Count */
  DCL BLOCK             CHAR(2560) ;
  DCL RECS(10)          CHAR(256) DEF BLOCK ;


  CALL THASDIR(SEGPDD,MEMBERS,N,#M) ;    /* Directory List */
  CALL THASRDS('OPEN','        ',RC,#REC,BLOCK) ;

  DO I = 1 TO #M ;
      CALL THASRDS('FIND',MEMBERS(I),RC,#REC,BLOCK) ;
      CALL THASRDS('READ',MEMBERS(I),RC,#REC,BLOCK) ;

      DO WHILE (RC = 0 ) ;

            DO J = 1 TO #REC ;
                  WRITE FILE (OUT) FROM (RECS(J)) ;
            END ;
            CALL THASRDS('READ',MEMBERS(I),RC,#REC,BLOCK) ;

      END ;
  END ;

  CALL THASRDS('CLOS','        ',RC,#REC,BLOCK) ;

//SEGPDD  DD  DSN=THASD.SEGPDS,DISP=SHR,
//        DCB=(RECFM=U,BLKSIZE=256)
//SEGPDS  DD  DSN=THASD.SEGPDS,DISP=SHR
//OUT     DD  DSN=&&TEMP,DISP=(NEW,PASS),
//        DCB=(RECFM=FB,LRECL=256,BLKSIZE=5120),
//        UNIT=SYSDA,SPACE=(TRK,(10,5),RLSE)
```

8.2.4 Program THAS203 - Select by Data Fields

```
Purpose:  to select accidents from a subset based upon the values in data
          fields.

Input:      - SYSIN     - selection criteria (from the menu)
            - INDATA    - input subset data
            - INDESC    - input subset description
            - FLDNAM    - HAS field data - THASP.TABLE(FLDNAMES)

Output:     - SDATA     - selected accident records
            - SDESC     - description file for SDATA
            - XDATA     - records excluded - records not selected
            - XDESC     - description file for XDATA
            - SYSPRINT  - selection criteria and record counts

Description:

This program uses the same "select by data field" code as program THAS200.
This program gets its input from an accident subset (instead of the PDS
Master).
```

8.2.5 Program THAS205 - Select counter-measure accident type.

```
Purpose:  to select accidents from a subset based upon the Counter-Measure
accident type.

Input:      - SYSIN     - selection criteria (from the menu)
                          line 1:  up to 13 accident types, format: nn nn nn
                          line 2:  2 bit flags for SDATA, XDATA output,
                                   in cols 1 and 3
            - INDATA    - input subset data
            - INDESC    - input subset description
            - ACCTYPS   - THASP.TABLE(ACCTYPES)
                        - accident type numbers and descriptions

Output:     - SDATA     - selected accident records
            - SDESC     - description file for SDATA
            - XDATA     - records excluded - records not selected
            - XDESC     - description file for XDATA
            - SYSPRINT  - selection criteria and record counts

Description:

Subroutine THASSAT is used to determine the set of counter-measure accident
types for each accident.  (An accident may fall into zero or more counter-
measure accident type categories.)  If any of the types in the set matches
any of the requested types, the accident is selected.
```

8.2.6 Program THAS210 - Accident-Prone Locations

Purpose: to identify and report Accident-Prone Locations. See the User's Manual section 6.5.1 for a description of terms and functions of this program.

Input:

- SYSIN - control parameters:
  - conditions defining "accident-prone"
  - location radius
  - landmark types
  - location types
  - costs
  - combined report output flag
- accident file output control:
  - two bit flags, controlling whether records included in acc-prone locations and records not in acc-prone locations should be written out to subset files SDESC and XDESC (respectively).
- output CSV file field names
- report sorts:
  - a sort order character (A or D), or blank, for each sort field. (See the example SYSIN file following.)

- INDATA - input accident data file
- INDESC - description of INDATA

- LANDMRK - LKI Landmark file.
- SHNFIL - Segment-Highway-Node file, read into the Highway-Segment-Node table in subroutine THASSNT.
- SEGDIST - LKI Segment-Districts file
- TLKIDST - LKI Districts file
- CRITRAT - Critical Rates file (for table lookup)
- AVERAT - Average Location Rates file
- RATIOS - Averate Location Accident Type Ratios
- SEGCLAS - sorted Segment Classifications file
- SEGVOL - Segment Volumes file
- SWARWTS - Severity-Weighted Accident Rate Weights

Output:

- SDATA - records included in acc-prone locations
- SDESC - description file for SDATA

- XDATA - records excluded - not in acc-prone locations
- XDESC - description file for XDESC

- SYSPRINT - messages, summary etc.
- SYSOUT - report file for the internal sorting routine
- REPFILE - accident-prone locations report file
- CSVOUT - comma-separated-values output file

- COMBIN - Combined Report records file (REPFILE without headings)
- COMDESC - Combined Report description file (INDESC plus input parameters to THAS210)

I/O

- SORTIN - accident-prone location report records,
- SORTOUT - with a four-byte sequence number appended.

- SORTWK01 \
- SORTWK02 -> temporary work files for the internal sorting routine
- SORTWK03 /

Description:

Read the control information from SYSIN.

Read the Landmark and Highway-Segment-Node files into memory.

Process the Accident Data:

- Read Accident Data, storing the required information from the accident records of the last Radius.
- If a specified Location or Landmark Type is encountered, sum the information required from the stored radius, then read ahead another radius, accumulating totals.
- If there is a conflict between a Location Type and a Landmark Type, the Landmark will take precedence.
- If there are multiple landmarks and/or location codes at one location, the 1st matching landmark is used, or the 1st location type.
- Mark and count Accident-Prone Locations which overlap a previous one. (Overlaps are only possible if Radius > 0.)
- For each Accident-Prone Location, write to a temporary file a report record with a sequence number appended.

The report layout is on a following page. THASCHA sets up the fields of the report records.

Sorting

For each sort order specified, sort the report file and write a report complete with page headings and the input file description information.

All sorts are done from within the PL/I program. Subroutine THASHST looks after the rank assigning and sorting for both THAS210 and THAS220.

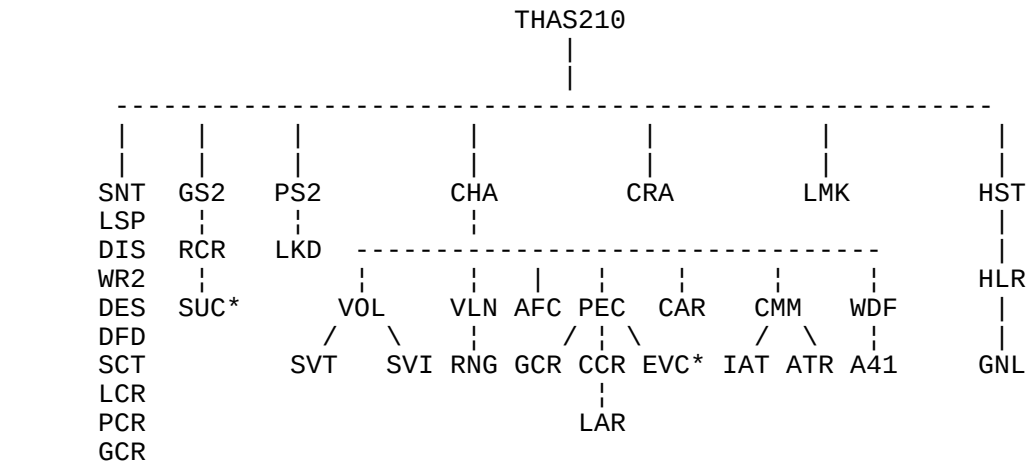
Combined Report

This is an obsolete feature.

When the Combined Report output flag is set, THAS210's report lines (without headings) are written to the COMBIN file, by subroutine THASHST. The non-printed SEQUENCE and SRCH_PATH_SEQ fields are included, and an 'L' is added at the end to identify the report record as a Accident-Prone Location record.

The input description file and THAS210's control parameters are copied to the Combined Report Description file, COMDESC.

THAS210 Structure Chart



(common with 220)

Utility: INL, ABT, PRC, DAT, T03, WDS, CAR, ESC, HCM, WDS, CIC, PDT, DMR, CCT

* SUC (SET_UP_CRITERIA) is an entry point in module EVC (EVALUATE_CRITERA). SUC defines the variable functions which are used each time EVC is called.

THAS210 Sample SYSIN File

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+

//SYSIN    DD *
This is a user-specified title.
'N' '>' '10' 'AND'           ; first condition
'S' '>=' '1.5' 'OR'          ; second condition
'R' '>'  'L'                 ; third condition
0.5                          ; level of significance for relational criteria
'AND'                        ; and/or counter-measure accident types
 1  9 10 13                  ; accident types
1.5                          ; level of sig. for counter-measure method
1.0                          ; radius
01 05 11                     ; landmark types
01                           ; location types
0 1                          ; output subset flags
1                            ; combined report output flag
0                            ; diagnosis file output flag
R                            ; weighted rate type to report
1                            ; add landmark descriptions flag
'RD'  'R '                   ; REGION, DISTRICT SORT AND BREAK
A           SEARCH SEQUENCE
            Hwy-Segment-Km
            Highway Class
            Average Daily Traffic (ADT)
            Accident Rate
            Road-Weighted Acc. Rate
            Rate/Critical Rate Ratio
            Road-Weighted Rate Ratio
```

```
            Number of Accidents
            Number of Vehicles
A           Number of Fatal Accidents
            Number of Fatalities
            No. of Injury Accidents
            Number of PDO Accidents
            Accident Severity Ratio
```

THAS210 Report Record Definition

The sections of the report line which are common to the Accident-Prone Locations and Accident-Prone Sections reports are kept in separate include files. This ensures that these sections **are** the same, and **remain** the same! (These report records were designed this way to make the **Combined Accident-Prone Locations and Sections Report** possible.)

Note that the Group ID in the printed record was added after this design was done. The Group ID is the same thing as SEQUENCE. It is appended to the printed lines at printing time.

The main include file is THASRHAL.

To see an up-to-date listing of THASRHAL with all its sub-include files expanded, see the compile listing of routine THASCHA, in THASD.LISTLIB(THASCHA).

8.2.7 Program THAS220 - Accident-Prone Sections

```
Purpose:    to identify and report Accident-Prone Sections of a user-
            specified length.  See the User's Manual section 6.5.2 for a
            description and definition of terms.

Input:      SYSIN    - control parameters:
                          - conditions defining "accident-prone"
                          - fixed section length
                          - costs
                          - combined report output flag
                       - accident file output control:
                          - two bit flags, controlling whether records included
                            in acc-prone sections and records not in acc-prone
                            sections should be written out to subset files
                            SDESC and XDESC (respectively).
                       - output CSV file field names
                       - report sorts:
                          - a sort order character (A or D), or a blank, for
                            each sort field.  (See the example SYSIN file
                            following.)

            INDATA   - input accident data file
            INDESC   - description of INDATA

            SHNFIL   - Segment-Highway-Node file, read into the
                       Highway-Segment-Node table in subroutine THASSNT.
            SEGDIST  - LKI Segment-Districts file
            TLKIDST  - LKI Districts file
            CRITRAT  - Critical Rates file (for table lookup)
            AVERAT   - Average Section Rates file
            RATIOS   - Averate Section Accident Type Ratios
            SEGCLAS  - sorted Segment Classifications file
            SEGVOL   - Segment Volumes file
            LMKFIL   - landmark file
            SWARWTS  - Severity-Weighted Accident Rate Weights


Output:     SDATA    - records included in accident-prone sections
            SDESC    - description file for SDATA

            XDATA    - records excluded - not in accident-prone sections
            XDESC    - description file for XDESC

            SYSPRINT - messages, summary etc.
            SYSOUT   - report file for the internal sorting routine
            REPFILE  - accident-prone sections report file.
            CSVOUT   - comma-separated-values file

            COMBIN   - Combined Report records file
            COMDESC  - Combined Report description file
                       (INDESC plus input parameters to THAS220)

 I/O        SORTIN   - accident-prone section report records,
            SORTOUT    with a four-byte sequence number appended.

            SORTWK01 \
            SORTWK02 -> temporary work files for the internal sorting routine
            SORTWK03 /
```

THAS220 Description:

The old term 'hazardous' is used here and in the code, for 'accident-prone'.

Accident records are stored in a circular buffer. The buffer has positions numbered 0 to BUFSIZE-1. To determine the location (P) in the buffer of a record, the following calculation is done:

```
P = MOD (record_number,BUFSIZE)
```

Thus if BUFSIZE is 1000, then records 1 to 999 go into locations 1 to 999, then record 1000 goes into location 0, record 1001 goes into location 1, etc. This means that records do not have to be shifted back in the buffer when old records are removed from the back end of the buffer.

The following RECORD NUMBER information is maintained in the program:

```
FR     -   first (oldest) record in the buffer.
BR,ER  -   begin and end points of the section currently being examined.
NR     -   NEXT accident (last one read into the buffer)
BLHS   -   Begin of the Last fixed Hazardous Section identified
ELHS   -   End of the Last fixed Hazardous Section identified
BFHS   -   Begin of the FIRST fixed Hazardous Section in the current set of
           overlapping fixed hazardous sections.
BWHS   -   Begin of the Worst fixed Hazardous Section in an extended
           hazardous section
EWHS   -   End of the Worst fixed Hazardous Section in an extended hazardous
           section
```

The 'pointer' equivalents of BR, ER, & NR (their locations in the buffer) are stored in BP, EP & NP.

In the following algorithm description, just B, E & N are used, for clarity. The following variables are also referred to:

```
FSLENGTH - fixed section length.
NAC      - number of accidents in a fixed section
WORST_NAC- number of accidents in the worst hazardous section
```

THAS220 Algorithm outline

```
-  Read the control information from SYSIN.
-  Read in the Highway-Segment-Node table from SHNFIL.
-  Read in the Search Path from the input description file (INDESC).
-  Read the first accident record into the storage array, and set B=1, N=1
   and E=0

DO until End of File:

   -  Read and store records until FSLENGTH past the begin accident (accident
      B), or until a discontinuity is encountered, incrementing E and N.
      (The Search Path is used to determine distances and discontinuities.)

   -  If section B to E is hazardous then:
            BLHS=B; ELHS=E;
            If BFHS=0 (ie if there are no haz. sections in waiting)
                  BFHS=B             (this is the First one)
                  WORST_NAC=NAC      (and so far is the Worst one)
                  BWHS=B; EWHS=E;
            ELSE          (there are haz sections in waiting)
                  If NAC > WORST_NAC then (save this one as the worst one)
                     WORST_NAC=NAC
                     BWHS=B; EWHS=E;
                  ENDIF
            ENDIF
      ENDIF

   Increment B until within FSLENGTH of the NEXT accident, or B=N

   If B is past the end of the last haz section (B > ELHS)
      Perform calculations for extended hazardous section BFHS to ELHS and
      write out a report record.  (If there were no overlapping fixed
      hazardous sections, the 'extended' hazardous section may not actually
      be extended, but equal a fixed section).

      Note that the end point of the hazardous section is FSLENGTH past the
      begin point of the last fixed length hazardous section, not at the last
      accident (ELHS).

      Set BFHS=0, and clear WORST FS variables.
   ENDIF
ENDDO
```

Section Lengths

Each KMMARK represents 0.1 KM of road, so a section with start Km of 4.0 and a stop of 5.0 represents 1.1 Km of road. Thus a 1.0 Km section of road with a
start Km of 4.0 must have a stop Km of 4.9. The actual section of road represented (if you want to get picky) is from 3.95 to 4.95. Thus the end-of-section KMMARK in the program (STOP_KM) is reduced by 0.1 on the report.

In the case where a discontinuity forces a short section, the KMMARK at the discontinutity is reported as the end-of-section, and 0.1 is ADDED to the length.

Sorting

The report file is first sorted on the designated 'rank-by' field, so that the RANK field can be filled. The rank-by field is currently Accident Rate, but this may become user-specifiable. For reports in order of the 'rank-by' field, no further sorting is required.

For each other sort order specified, sort the report file and write a report complete with page headings and the input file description information.

All sorts are done from within the PL/I program. Subroutine THASHST looks after the rank assigning and sorting for both THAS210 and THAS220.

Extended Hazardous Sections

When there are overlapping fixed-length hazardous sections, they form an extended hazardous section. In this case, the report contains a line for the extended section, followed by a line for the worst fixed-length hazardous section within the extended section, followed by a blank line. The format of the worst-hazardous-section lines is identical to the format of the extended-hazardous-section lines.

The report layout is on the following page. THASPHS calls THASFHS (either once or twice) to set up the fields of the report records.

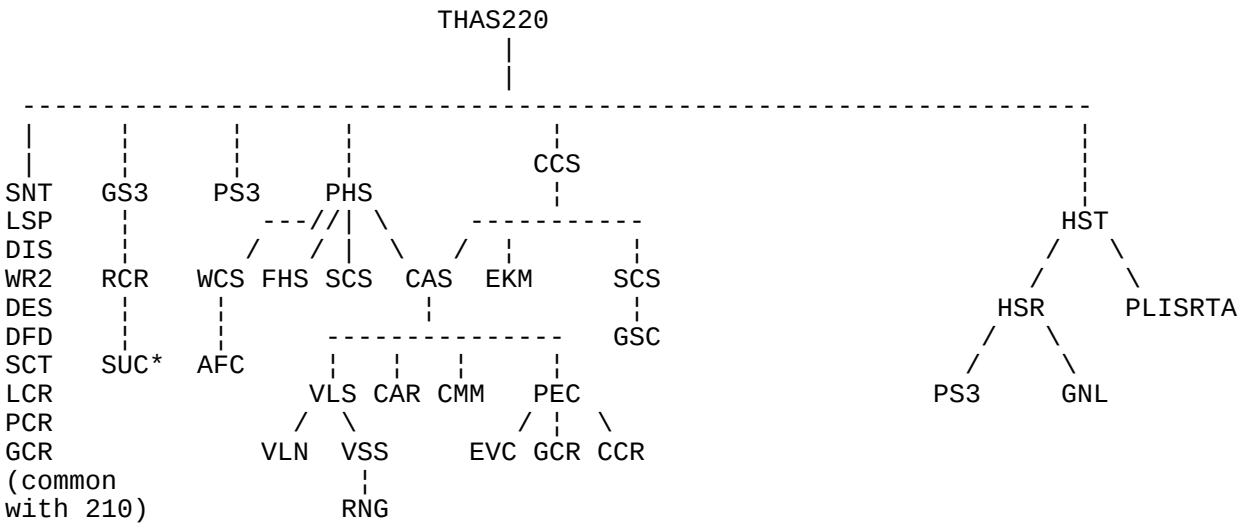
Combined Report

This is an obsolete feature.

When the Combined Report output flag is set, THAS220's extended-section report lines (without headings) are written to the COMBIN file. The non-printed SEQUENCE and SRCH_PATH_SEQ fields are included, and an 'S' is added at the end to identify the report record as a Hazardous Section record.

The input description file and THAS220's control parameters are copied to the Combined Report Description file, COMDESC.

THAS220 Structure Chart



Utility: INL, ABT, PRC, DAT, T03, WDS, CAR, ESC

* SUC (SET_UP_CRITERIA) is an entry point in module EVC (EVALUATE_CRITERA). SUC defines the variable functions which are used each time EVC is called.

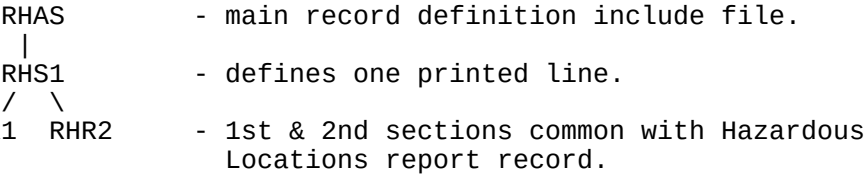
THAS220 Report Record Definition

The sections of the report line which are common to the Hazardous Location and Hazardous Section reports are kept in separate include files. This ensures that these sections **are** the same, and **remain** the same! (These report records were designed this way to make the **Combined Hazardous Locations and Sections Report** possible.)

In the case of extended hazardous sections, there are two printed lines on the report: one for the extended section, and one for the worst hazardous section within the extended one. Since these lines must be kept together while sorting, they are written as one long record.

The main include file THASRHAS is used in subroutine THASPHS.

The include files are nested as follows:



To see an up-to-date listing of THASRHAS with all its sub-include files expanded, see the compile listing of routine THASPHSA, in THASD.LISTLIB(THASPHS).

8.2.8 Program THAS225 - Specified Section Analysis

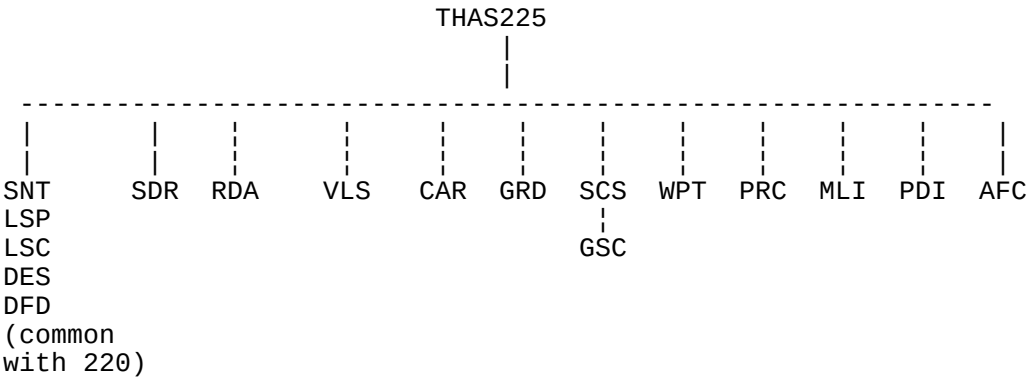
```
Purpose:    Produces accident statistics for specified sections of highway.
            Sections are specifed in a user-prepared CSV file.  Statistics
            are the same as those calculated by THAS210 and THAS220.

Input:      INDATA      - input accident data file
            INDESC      - description of INDATA
            DATELIM     - THASP.PDSMAST.DATELIMS
            SECDEF      - CSV file of section definitions
            TLKIDST     - District file
            LMKFIL      - Landmark file
            SHNFIL      - Segment-Highway-Node file, read into the
                          Highway-Segment-Node table in subroutine THASSNT.
            CRITRAT     - Critical accident Rates file
            AVERAT      - Section average accident rates file
            AVERAT2     - Location average accident rates file
            SEGCLAS     - Segment-Class file
            SEGDIST     - Segment-District file
            SEGVOL      - Segment Volume file
            SYSIN       - title
            SWARWTS  - Severity-Weighted Accident Rate Weights


Output:     SDATA    - records included in one or more specified section.
                     - (optionally duplicated if in > 1 section)
            SDESC    - description file for SDATA

            XDATA    - records excluded - not in an specified section
            XDESC    - description file for XDESC

            STATS    - CSV output file of statistics.
                     - fields and order of fields are selectable by the user

            SYSPRINT - messages, summary etc.


Description:

Both the input section definitions CSV file and the output statistics CSV
file are highly cusomizable.  Fields are identified and specified by field
name.  A list of the required output field names, and their order is read
from SYSIN.  Fields within the input section definitions file are located by
their field names.

The program reads accidents and updates counters for the sections in which
each accident falls.  Then the same subroutines as used by THAS210 and
THAS220 are used to do Location or Section statistical calculations.  By
default, Section calculations are done, but Location calculations may be
specified in with each section definition.


 In the program, each defined "section" is considered as 1 or more
 sub-sections.  Each node (segment connection) and each non-node part
 of each segment is considered a sub-section.

   R = N/(LV)
    where R is acc rate,
          N is total number of accidents over the time period.
          L is total length,
          V is total number of vehicles over the time period
```

```
  Over M sub-sections, with an index i
         N = Sum(Ni)
         L = Sum(Li)
         V = Sum(LiVi)/Sum(Li)    (i.e. volume is weighted by
                                   each sub-section's length)
```

When "IncOpp" is specified, opposite sections are added to the list of sub-sections, so their lengths are added to the total length.

What happens in effect, when an equal-length opposite section is included is: the length is doubled and the volume is averaged (halved), so you end up with the same rate.

THAS225 Structure Chart



Sample SYSIN file:

```
0
1
0 0                          ; OUTPUT SUBSET FLAGS
.5
ID
Seg1
Km1
Seg2
Km2
Length
Date1
Date2
Class
CritRate Calc
ADT
ADT Src
Volume
Rate
RW Rate
ASR
CritRate Lkup
LmkType
LmkType Src
Loc/Sec
#Acc
#Veh
Fat#Acc
Fat#Fat
Fat#Inj
Inj#Acc
Inj#Inj
PDO#Acc
Note
END
/*
```

8.2.9 Program THAS230 - Histogram Report

```
Purpose:    to print a histogram of the numbers of fatal, injury, and PDO
            accidents at each location of the selected data.

Input:      INDATA      - input accident data file
            INDESC      - description of INDATA
            TLKISEG     - LKI Segment file
            LMKFIL      - LKI Landmark file
            SHNFIL      - Segment-Highway-Node file, read into the
                          Highway-Segment-Node table in subroutine THASSNT.
            SYSIN       - title
                        - # KMMARKs per histogram line
                        - # fatal accidents per F in the histogram
                        - # injury accidents per I in the histogram
                        - # PDO accidents per P in the histogram

Output:     SYSPRINT    - messages, description file, printed histogram

I/O:        SMRYFIL     - temporary file containing a summary of number of
                          accidents of each type at each location
```

Description:

The number of accidents of each type (fatal, injury, and property-damage-only) at each location are counted and saved in the summary file (SMRYFIL). Locations with no accidents are not shown in SMRYFIL.

The summary file is read again to produce and print the actual lines of the histogram report.

Each data line of the histogram may display either all accidents at a node, or all accidents for one KMMARK or KMMARK-range of a segment. For example, a particular line might contain all the accidents for KMMARK 3.2 of a segment, or it might contain all the accidents for KMMARKs 7.1-8.0 of a segment. In the first case, the user asked that each histogram line represent only 0.1 kilometers; in the second case, the user asked that each line represent a full kilometer, or ten KMMARKs.

Each symbol in the histogram (F, I, or P) may represent one or more accidents, depending on what the user requested. The number of accidents per symbol may be the same for all types of accidents (fatal, injury, PDO) or it may be different for each type.

The Search Path (from INDESC) may have discontinuities in it. These search path discontinuities are not necessarily the same as discontinuous nodes; rather, they are points where the requested list of locations has a gap in it, such as when a Histogram Report is generated for segments from two different highways. Search path discontinuities are signalled by a row of dashes in the Search Path in INDESC.

Nodes, the beginnings of segments, and search path discontinuities are each signalled by a label line at the appropriate place within the Histogram Report.

Multiplier-factor descriptions in the headings of each page:

Among the headings on each page of the histogram is a line that describes how many accidents are represented by each symbol (F, I, or P) in the histogram. It may have one of several different forms.

If each symbol represents only one accident, the line appears as:
No multiplier factor. Each F, I, or P represents one fatal, injury, or PDO accident.

If all symbols represent the same number of accidents (greater than one), the line appears as:
Multiplier: Each F, I, or P represents up to NNN fatal, injury, or PDO accidents.
(where NNN stands for the number of accidents per symbol).

If the three symbols represent different numbers of accidents, the line appears as:
Multipliers: Each F = up to NNN fatal accidents. Each I = up to NNN injury accidents. Each P = up to NNN PDO accidents.
If a particular symbol represents only one accident, the words 'up to' will be eliminated for that symbol (and 'accidents' will become 'accident').

Location descriptions at the bottom of each page:

If a page of the histogram contained any part of a segment (excluding nodes and discontinuities), then there will be lines given at the bottom of the page describing the segment-locations covered on the page. An example of these descriptive lines would be:

```
Histogram for:  HWY 1A (BRITISH COLUMBIA ; OLD ISLAND ; CHEMAINUS)
                SEGMENT 0303 (VICTORIA - GOLDSTREAM)   FROM KM  12.5 TO KM  15.3
```

The descriptions in parentheses are taken from the highway and segment descriptions in the LKI Highway and Segment files (THASP.HIGHWAY and THASP.SEGMENT).

Algorithm outline

Do all initialization:

- Check for an empty accident file.
- Read the Segment-Highway-Node table from SHNFIL.
- Read the Search Path from INDESC.
- Remove nodes from the segments within the Search Path. (Nodes will have their own separate entries in the Histogram Report.)
- Read the input parameters from SYSIN.
- Read INDATA and create the summary file SMRYFIL.
- Read and print out the Description File INDESC as the first page(s) of the report.
- Print the header of the first page of the Histogram.
- Set all accident totals for nodes and segment to zero.

Print the histogram from the summary file:

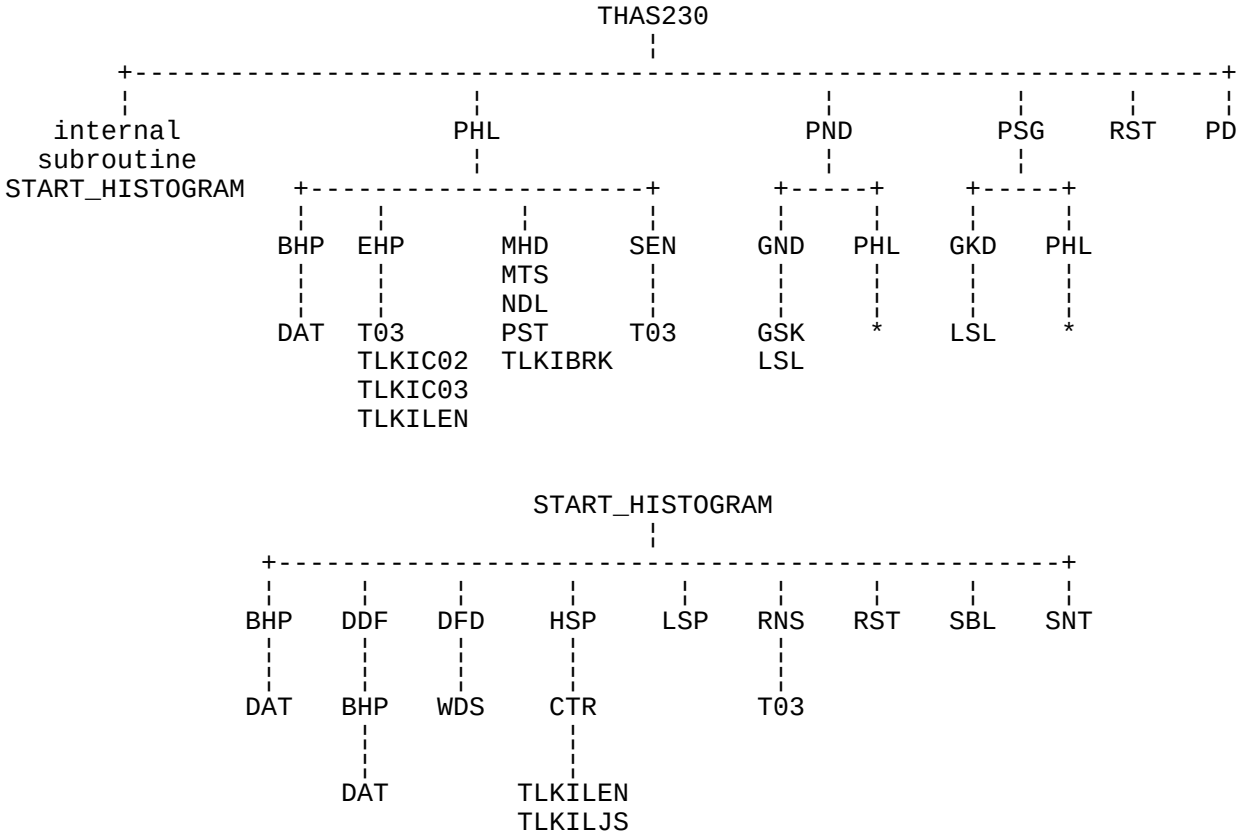
- For each entry in the Search Path:
  If this entry is for a discontinuity, then:
    - Print a discontinuity-marker line in the histogram.

  If this entry is for a node, then:
    - Determine the accident totals for the node from the SMRYFIL.
    - Print the label line, histogram, and accident totals for the node.
    - Reset the node totals to zero.

  If this entry is for a segment (or part of one), then:
    - Do while the current summary record belongs to this segment:
      - Save the totals from the summary line in the segment totals array. (This array has an entry for each possible KMMARK from 0.0 to 999.9 of a segment. Thus, the histogram for an entire segment can be printed at once.)
      - Read another summary record from the SMRYFIL.
    - Print the label line for this segment.
    - Print a histogram and accident totals for each KMMARK or KMMARK-range in the segment.
    - Reset the segment totals array to zero.

Print the final 'HISTOGRAM COMPLETE' line.

THAS230 Structure Charts



* Structure charts for these subroutines are shown (once only) elsewhere on the page.

Utility: ABT

Histogram Report Layout

```
        1         2         3         4         5         6         7         8         9        10        11        12        13
1234567890    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    012


                                    *** H.A.S.  A C C I D E N T   H I S T O G R A M ***                          PAGE zzz9
                                                  Version  2.1
Data Selection Dates:  yymmdd - yymmdd                                                  FAT: # FATAL ACCIDENTS
Report date:  yy/mm/dd     hh:mm                                                        INJ: # NON-FATAL INJURY ACCIDENTS
                                                                                        PDO: # PROPERTY-DAMAGE-ONLY ACCIDENTS
                                        User-defined title                              TOT: TOTAL ACCIDENTS (FAT, INJ & PDO)
[Multiplier-factor line]

=============================================================================================================================================
                                                                                                                    ¦   C U M U L A T I V E
  LANDMARK DESCRIPTION  KMMARK¦              HISTOGRAM OF ACCIDENT FREQUENCY                FAT INJ PDO TOT ¦  FAT   INJ   PDO   TOT
---------------------------------------------------------------------------------------------------------------------------------------------
                             ¦                                                                              ¦
NODE 99999999                ¦                                                                              ¦
landmark description of node ¦ FFFFIIIIIIIIPPPPPPPPPPPPPPPPP                                zz9 zz9 zz9 zz9 ¦zzzz9 zzzz9 zzzz9 zzzz9
                             ¦                                                                              ¦
SEGMENT 9999                 ¦                                                                              ¦
landmark description    zz9.9 ¦ FFIIIPPPP                                                    zz9 zz9 zz9 zz9 ¦zzzz9 zzzz9 zzzz9 zzzz9
landmark description    zz9.9 ¦ FFIIIPPPP                                                    zz9 zz9 zz9 zz9 ¦zzzz9 zzzz9 zzzz9 zzzz9
landmark description    zz9.9x¦ FFIIIPPPP                                                    zz9 zz9 zz9 zz9 ¦zzzz9 zzzz9 zzzz9 zzzz9
landmark description    zz9.9x¦ FFIIIPPPP                                                    zz9 zz9 zz9 zz9 ¦zzzz9 zzzz9 zzzz9 zzzz9
                             ¦                                                                              ¦
                             ¦                                                            FAT INJ PDO TOT ¦
                             ¦                                                                            ¦
                             ¦                        Accident totals for segment 9999:   zz9 zz9 zz9 zz9 ¦
                             ¦                                                                              ¦
NODE 99999999                ¦                                                                              ¦
landmark description of node ¦ FFFFIIIIIIIIPPPPPPPPPPPPPPPPP                                zz9 zz9 zz9 zz9 ¦zzzz9 zzzz9 zzzz9 zzzz9
                             ¦                                                                              ¦
                             ¦ --------------- DISCONTINUITY IN SEARCH PATH ---------------                  ¦
                             ¦                                                                              ¦
                             -+----+----+----+----+----+----+----+----+----+----+----+----+
                              0        10        20        30        40        50        60
                                                                                    FAT INJ PDO TOT
                                                                                   -----------------
                                                                Accident totals this page:   zz9 zz9 zz9 zz9


1234567890    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    012
```

8.2.10 Program THAS232 - Fatal, Injury, PDO Accident Counts

```
Purpose: Create a comma-delimited, text (down-loadable) file containing
         the number of Fatal, Injury, PDO and total number of accidents
         at each location with accidents.

Input:      INDATA      - input accident data file
            INDESC      - description of INDATA
            SYSIN       - 1st record: a title, no quotes
                        - 2nd record: a 1 or a zero (1 to put heading
                          information at the top of the output file.

Output:     SYSPRINT    - messages, summary, etc.
            OUTFIL      - output CSV file.


Description:

A user title, and an INTRO flag are read from SYSIN.

If the INTRO flag is TRUE, program information and the description file of
the input subset are put first in the output file.

The SYSPRINT dataset contains only summary information: the number of
records/accidents read, and the total number of fatal, injury and PDO
accidents.

The summary information is also written to the end of the output file.

Nodes are NOT identified as such.  All accidents at a node is labelled with
the first seg-km of the node encountered in the data.
```

8.2.11 Program THAS240 - Details Report

Purpose: to print a full description of each accident record in easy-to-read tabular form.

Input:
- INDATA - input accident data file
- INDESC - description of INDATA
- VSAMLTX - location text (Place, On, At)

Output:
- SYSPRINT - messages, summary, etc.

Description:

Full descriptions of MV104-form terms are read in initially into an array, partitioned into their corresponding table numbers. Each code from the accident record is searched sequentially using the table number array and then again using the positional value returned on the whole array. A table look-up routine returns a 40-char descriptor for each code. Each code is checked to see if it is within the effective date of the code.

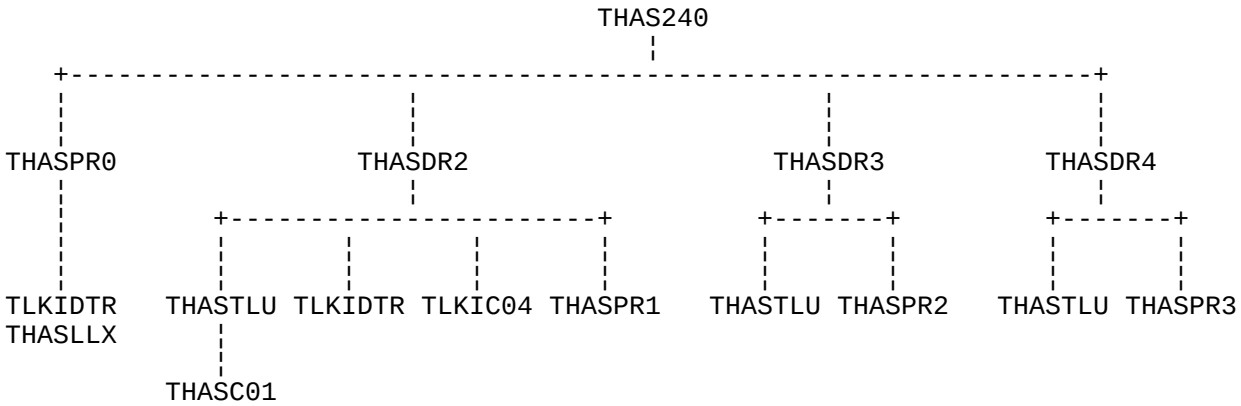
Algorithm outline

- Read the INDESC file and print it out.
- Read the MV104 file into the MV104 table and print each record on a page.
- Read the INDATA file and print each record on a separate page.

DO until End of File:

- Read each record.
- Print out header information.
- Look up accident information and print it out.
- Look up vehicle information and print it out.
- Look up victim information and print it out.

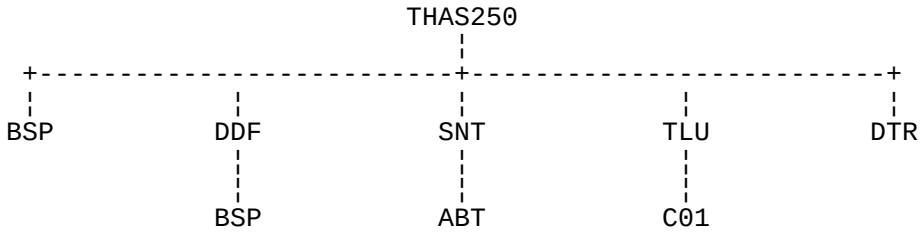
THAS240 Structure Chart



8.2.12 Program THAS250 - Summary Report Version 1

```
Purpose:    to produce a printed report showing some useful fields of
            accident records, one line per accident.

Input:      INDATA      - input accident data file
            INDESC      - description of INDATA
            SHNFIL      - Segment-Highway-Node file, read into the
                          Highway-Segment-Node table in subroutine THASSNT.

Output:     SYSPRINT    - messages and printed report

Description:

-  Read the Highway-Segment-Node table from SHNFIL.

-  Read the description file INDESC and print it out.

-  Read the first page-1 accident record.

-  Do while not end of file:

   -  Convert the accident fields as necessary and transfer them to the
      output summary record.  The fields transferred are:
            ACCASE, ACCDATE, LOCN_CODE, TOTALKLD, TOTALINJ, TOTALVEH,
            DIAGRAM, LOCN_TYPE, ROADSURF, WEATHER, LIGHTING, VEHDIR1, VEHDIR2

   -  If the accident occurred at a node, then:
         -  Insert a node marker (n) between the highway and segment in the
            location code.

   -  If the accident occurred at an obsolete location, then:
         -  Insert an obsolete location marker (x) between the segment and
            KMMARK in the location code.

   -  Determine whether the accident was a fatal, injury, or property-damage-
      only accident and set the accident type in the summary record
      accordingly, to FAT, INJ, or PDO.

   -  If this accident is the first one occurring at a particular node, then:
         -  Print a blank line and a line identifying the node number.

   -  If this accident is the first one occurring in a particular segment,
      then:
         -  Print a blank line and a line identifying the segment number.

   -  Print the summary record.

   -  Save the identifying data (node #, segment #, and atnode flag) from the
      current accident.  This is the data used to determine whether another
      accident is the first one occurring at a particular node or segment.

   -  Read another page-1 accident record.

(End Do while not end of file)
```

THAS250 Structure Chart



THAS250 Summary Report Layout

```
        1         2         3         4         5         6         7         8         9        10        11        12        13
1234567890    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    012
```

***** H.A.S. A C C I D E N T S U M M A R Y R E P O R T *****

Report Date: yr/mo/da hr:mn Page ZZ9

Accident Case #	Accident Date	Location Hwy Sgmt Km	Acc. Type	Tot Kld	Tot Inj	Tot Veh	Primary Occurrence	Accident Location	Road Surf. Cond.	Weather Conditions	Lighting Condition	Drectn Travel V1	Drectn Travel V2
AAAAAAAA	da MON yr	ZZ9An9999xZZ9.9	AAA	ZZ9	ZZ9	ZZ9	description	description	descr	description	descript.	A	A

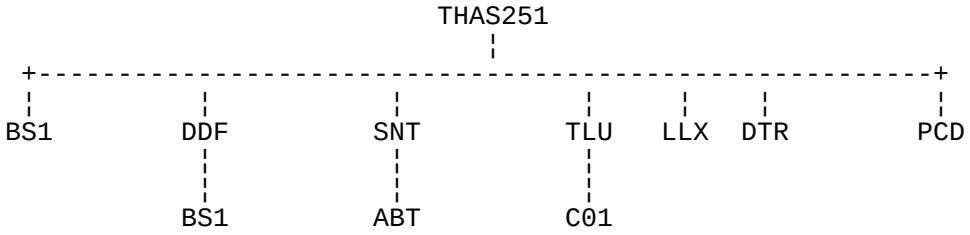
```
1234567890    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    012
        1         2         3         4         5         6         7         8         9        10        11        12        13
```

8.2.13 Program THAS251 - Summary Report Version 2

```
Purpose:    to produce a printed report showing some useful fields of
            accident records, one line per accident.

Input:      INDATA      - input accident data file
            INDESC      - description of INDATA
            SHNFIL      - Segment-Highway-Node file, read into the
                          Highway-Segment-Node table in subroutine THASSNT.
            VSAMLTX     - Location Text

Output:     SYSPRINT    - messages and printed report

Description:

-  Read the Highway-Segment-Node table from SHNFIL.

-  Read the description file INDESC and print it out.

-  Read the first page-1 accident record.

-  Do while not end of file:

   -  Convert the accident fields as necessary and transfer them to the
      output summary record.  The fields transferred are:
            ACCASE, ACCDATE, ACCHOUR, LOCN_CODE, TOTALKLD, TOTALINJ,
            TOTALVEH, DIAGRAM, LOCN_TYPE, CONTRB11, ROADSURF, WEATHER,
            VEHDIR1, VEHDIR2
      (LOCN_TYPE is left as a code to give room for CONTRB11 in full detail.
      The meanings of the codes are given at the bottom of the report.)

   -  If the accident occurred at a node, then:
         -  Insert a node marker (n) between the highway and segment in the
            location code.

   -  If the accident occurred at an obsolete location, then:
         -  Insert an obsolete location marker (x) between the segment and
            KMMARK in the location code.

   -  Determine whether the accident was a fatal, injury, or property-damage-
      only accident and set the accident type in the summary record
      accordingly, to FAT, INJ, or PDO.

   -  If this accident is the first one occurring at a particular node, then:
         -  Print a blank line and a line identifying the node number.

   -  If this accident is the first one occurring in a particular segment,
      then:
         -  Print a blank line and a line identifying the segment number.

   -  Print the summary record.

   -  Save the identifying data (node #, segment #, and atnode flag) from the
      current accident.  This is the data used to determine whether another
      accident is the first one occurring at a particular node or segment.

   -  Read another page-1 accident record.

(End Do while not end of file)

-  Print out the meanings of the codes used in the report.  These are the
```

node markers, the obsolete location markers, and the LOCN_TYPE codes.

THAS251 Structure Chart



THAS251 Summary Report Layout

```
        1         2         3         4         5         6         7         8         9        10        11        12        13
1234567890    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    012

                              ***** H.A.S. A C C I D E N T  S U M M A R Y  R E P O R T *****

Report Date:  yr/mo/da   hr:mn                                                                                          Page ZZ9

                                                                                                                 ¦Road ¦           ¦Drctn
Accident¦ Accident¦     ¦    Location    ¦Acc¦Tot¦Tot¦Tot¦    Primary     ¦Acc.¦  First Contributing Factor  ¦Surf.¦  Weather  ¦Travl
 Case # ¦   Date  ¦ Time¦ Hwy Sgmt   Km ¦Typ¦Kld¦Inj¦Veh¦   Occurrence   ¦Loc.¦      for First Vehicle      ¦Cond.¦Conditions ¦V1 V2
--------¦---------¦-----¦---------------¦---¦---¦---¦---¦----------------¦----¦-----------------------------¦-----¦-----------¦-----
AAAAAAAA¦da MON yr¦99:99¦ZZ9An9999xZZ9.9¦AAA¦ZZ9¦ZZ9¦ZZ9¦description     ¦ 99 ¦description                ¦descr¦description¦ A  A


1234567890    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    012
        1         2         3         4         5         6         7         8         9        10        11        12        13
```

8.2.14 Program THAS260 - Rate Table

```
Purpose:  to produce a table of accident rates by Highway Classification
          and Traffic Volume.

Input:    SYSIN     - Title,
                    - accident rate type (SECTION, LOCATION, or LANDMARK)
                    - up to 10 2-character landmark types
                    - flags for optional cell contents.
                    - flags to request Class-Rate report, and type of sorting
                      done on it
                    - flag to request long sections be divided into shorter
                      lengths
                    - Section lengths (highway class and maximum length)
                    - Column definitions (classes in each column)
                    - Row definitions (volume ranges in each row)
          SEGCLAS   - Segment-Class file, sorted by segment search
                      sequence.
          CLASNAM   - THASP.CLASS.NAMES (opened by subroutine THASCLN)
          INDATA    - input accident data file (also in search sequence)
          INDESC    - description file for INDATA
          SEGVOL    - one of the segment volumes files: THASP.SEGVOL1,
                      THASP.SEGVOL2, or THASP.SEGVOL3
          NODEVOL   - node volume file THASP.NODEVOL
          SHNFIL    - Segment-Highway-Node file
          TLKIDST   - THASP.DISTRICT (LKI file)
          SEGDIST   - THASP.SEGDIST (LKI file)
          LMKFIL    - THASP.LANDMARK (LKI file)
          SWARWTS   - Severity-Weighted Accident Rate Weights


Output:   SYSPRINT  - messages, summary etc.
          REPFILE   - Rate Table report
          REPFIL2   - Class-Rate report

          SYSOUT    - report file for the internal sorting routine

 I/O      SORTIN    - class-rate report records, unsorted
          SORTOUT   - class-rate report records, sorted

          SORTWK01  \
          SORTWK02  -> temporary work files for the internal sorting routine
          SORTWK03  /
```

8.2.14.1 Sorting

The Class-Rate report is sorted in one of two ways: by segment and KMMARK (sort type is SEGMENT) or by highway class, ADT, segment, and KMMARK (sort type is CLASS).

The Class-Rate report may also be left unsorted, with the Class-Rate records left in the order generated by THAS260, i.e. primarily in segment-KMMARK order but with the highway sections that fell in no cell of the Rate Table at the end of the report. (These are usually sections with no available class or volume, but may also be sections that did not correspond to any defined cell.) This option is chosen by a blank sort type.

8.2.14.2 Notes on SCLTAB and HSECTAB

These tables are the Seg-Class Table (SCLTAB) and the Highway Sections Table (HSECTAB).

These two tables can easily be confused, since they contain similar data. However, they should be distinguished, since they **lack** different subsets of the provincial highway system.

The SCLTAB contains all the data from the SEGCLASS file (slightly modified in format). It contains all highway sections in the province with a defined segment classification.

The HSECTAB contains only the highway sections that are on the Search Path. This varies from one run of the Rate Table to the next. Each entry in the Search Path is split into one or more HSECTAB entries, each of which has a single segment classification. The HSECTAB may, therefore, be a subset of the SCLTAB. However, if the Rate Table is run on the whole province, the HSECTAB will include whatever sections have not yet had a classification defined in the SEGCLASS file; in this case, the HSECTAB includes all of the SCLTAB, plus some.

The SCLTAB **lacks** whatever highway sections have not yet had their classifications determined and added to the SEGCLASS file. The HSECTAB **lacks** everything that is not on the current Search Path.

Routines that use these two tables in the Rate Table program are:

- THASLSC - Reads in the SCLTAB from the SEGCLASS file, and resets the end-KMMARKs of sections to the actual segment lengths when necessary. (Some sections in the file end with a KMMARK of 999.9.)
- THASDSS - Divides the sections in SCLTAB into smaller pieces, if this options is requested (SECTION-type Rate Tables only). The smaller pieces are still in SCLTAB, so the original undivided SCLTAB is lost.
- THASGSC (used by THASFST & THASCEL) - Finds a section in the SCLTAB containing a particular location (returns START_KM, END_KM, CLASSIFICATION).
- THASFST - Reads SCLTAB and create a single entry in HSECTAB.
- THASSPS - Goes through the Search Path. For each node or segment in the Path, THASSPS calls THASFST as many times as necessary to add entries to HSECTAB. Note that one Search Path section may have several different highway classes, and may therefore become several HSECTAB sections.
- THASSCD - Gets data from SCLTAB via calls to THASCEL and uses that data (SECTION_START_KM, SECTION_END_KM, SECTION_CLASSIFICATION) in its

own processing. Uses HSECTAB only as a parameter in calls to THASCOS and THASIUS.

THASLAD - Calls THASCEL, which gets data from SCLTAB, but only uses classification (not start-km or end-km).

THASLLD - As in THASLAD.

THASCOS - Finds entry in HSECTAB containing a particular section and marks it as completed. (SECTION-type Rate Tables only.) A section will only be passed to THASCOS if it contains accidents, and has a highway class and traffic volume.

THASIUS - Goes through HSECTAB, finding all entries not marked by THASCOS, and performs equivalent processing on them to the work done by THASSCD. (SECTION-type Rate Tables only.)

THASTCL - Uses HSECTAB to determine length of highway in each Rate Table cell. Calls THASCEL, but tells it not to get data from SCLTAB (THASCEL will use the HSECTAB data instead.)

8.2.14.3 Algorithm Outline

Define 5 arrays (currently maxrow = 100 and maxcol = 9):

A(maxrow,maxcol)	-	number of accidents for each Class-Volume cell
FAT(maxrow,maxcol)	-	number of fatal accidents for each Class-Volume
RWACC(maxrow,maxcol)	-	sum of road-weight percentages for all accidents in each Class-Volume cell
RWFAT(maxrow,maxcol)	-	sum of road-weight percentages for all fatal accidents in each Class-Volume cell
LMV(maxrow,maxcol)	-	length * #vehicles or just #vehicles for each Class-Volume cell

indexed by i - row number, and j - column number.

1. Create the Highway Sections Table, as follows:
   Note that each entry in the Highway Sections table (HSECTAB) is part or all of a SINGLE entry in the SEGCLASS table (SCLTAB). I.e., HSECTAB entries do not span SEGCLASS sections (or the smaller SEGCLASS sections created by THASDSS).

   - THASLSC: Load the SEGCLASS table (SCLTAB).
   - IF the user wants small sections, then:
     - THASDSS: Divide the SEGCLASS table into smaller pieces.
   - THASLSP: Load the Search Path (PATH).
   - THASRNS: Modify the Search Path table (PATH) so nodes are not included as part of section-type entries.
   - THASSPS (Creates the Highway Sections Table):
     FOR every entry in the Search Path:
       IF it's a segment-type entry,
         Add one or more segment-sections to the HSECTAB.
       ELSE IF it's a node-type entry,
         Add a node-section to the HSECTAB.

2. Read the first accident record.

+> 3. Find the location of the current accident in the SEGCLASS file, then,
¦ - extract the classification information.
¦ - determine which column definitions include this classification, and record the column number(s) j.
¦ - Call THASVOL to get the traffic volume (V) and ADT for the segment.
¦ - Determine which volume ranges the ADT is in, and record the row number(s) i.
¦

4. Read and count:
   - the number of accidents,
   - the number of fatal, injury, and PDO accidents,
   - the number of accidents that fall in one, more than one, or none of the Class-Volume cells,

   and accumulate:
   - road weights of all accidents (in RWACC),
   - road weights of all fatal accidents (in RWFAT),

   until the KMMARK exceeds the END-KM in the current SEGCLASS record, the segment changes, or the end of the accident file is reached.

5. If the rate type is SECTION, then:
   - Subtract START_KM from END_KM to get the section length (L).
   - for each qualifying row number i and column number j, add L*V to LMV(i,j).

   else (the rate type is LOCATION or LANDMARK):
   - for each qualifying row number i and column number j, add V to LMV(i,j).

6. Calculate the accident and fatality rates for the section just ended:

   ACRATE = (A/LV) * 1E6 RWACRATE = (RWACC/LV) * 1E6
   FTRATE = (FAT/LV) * 1E8 RWFTRATE = (RWACC/LV) * 1E8

7. If statistics were requested:
   - accumulate the sum and sum-of-squares of accident rates for each cell containing this section or location.

8. If the Class-Rate Report has been requested:
   - write a record to the Class-Rate Report sort file.

9. Add the number of accidents to A(i,j), and the number of fatal accidents to FAT(i,j).

10. If the end of the accident file has not been reached, return to step 2.

11. If the rate type is SECTION, add L*V to LMV(i,j) for each section in the search path without accidents, and for each qualifying i and j.

12. Calculate the accident rates for each element of the table:

```
    RATE(i,j) =     (A(i,j) / LMV(i,j)) * 1E6
  FATRATE(i,j) =   (FAT(i,j) / LMV(i,j)) * 1E8
   RWRATE(i,j) = (RWACC(i,j) / LMV(i,j)) * 1E6
RWFATRATE(i,j) = (RWFAT(i,j) / LMV(i,j)) * 1E8
```

13. Print the table, with title, headings, Rate Table type, and whatever information has been requested for each Class-Volume cell:
   - accident rate (always present),
   - fatal accident rate,
   - road-weighted accident rate,
   - road-weighted fatal accident rates,
   - statistics (supplied by subroutine THASCST),
   - percent of FAT/INJ/PDO accidents,
   - number of FAT/INJ/PDO accidents,
   - total number of accidents,
   - total highway length,

   and Classification Set translations (supplied by subroutine THASTCS).

14. If the Class-Rate Report was requested:
   - sort and print the Class-Rate Report.

LOCATIONS-AT-LANDMARKS Algorithm:

The LANDMARK-type rate table differs from the LOCATION-type table in that it is driven by the LKI landmark file instead of by the input accident data. Landmarks without accidents are included in the table (A and FAT are zero), but accidents not at landmarks, and accidents at landmarks not of specified types, are ignored. Each included landmark is otherwise treated identically to a location in the LOCATION-type rate table.

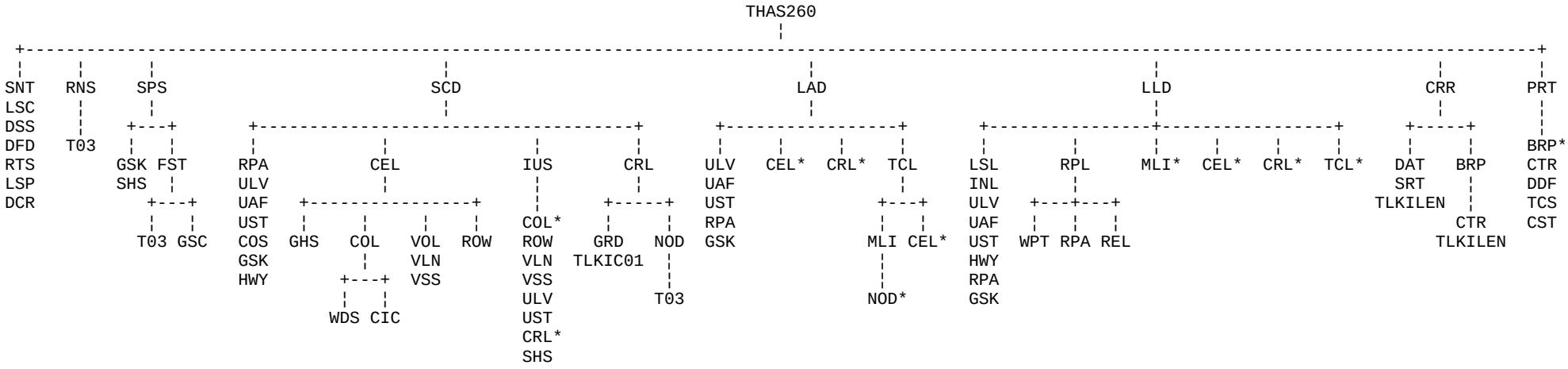
Highway Classification (Column) Definitions:

The method of defining a set of highway classifications is described in section 2.4.6 of the User's Manual (**Highway Classification Sets**). Example: '* CF'.

Example SYSIN file (designed to be read mostly with GET SKIP LIST statements):

```
'TITLE:'
This is the title.
'LANDMARK'              - SECTION, LOCATION, or LANDMARK (rate type)
A1 A2 A3                - Up to 10 landmark types
'STATS'                 - STATS or NOSTATS (statistics)
'FATRATE'               - FATRATE or NOFATRATE (fatal accident rate)
'RWRATE'                - RWRATE or NORWRATE (road-weighted accident
                          rate)
'RWFATRATE'             - RWFATRATE or NORWFATRATE (R-W fatal acc. rate)
'TYPPCT'                - TYPPCT or NOTYPPCT (% FAT/INJ/PDO)
'TYPNUM'                - TYPNUM or NOTYPNUM (# FAT/INJ/PDO)
'TOTACC'                - TOTACC or NOTOTACC (# accidents)
'NOLENGTH'              - LENGTH or NOLENGTH (kms. in cells)
'REPORT'  'SEGMENT'     - REPORT or NOREPORT (class-rate)
                            - CLASS, SEGMENT, or blank (sort type)
'DIVSECT'               - DIVSECT or NODIVSECT (divide sections)
'MAXLENGTHS:'           - Maximum section lengths
'U'   3.0               - Urban: 3.0 km sections
'R'  20.0               - Rural: 20.0 km sections
'X' 999.9               - Unnumbered (X-type): any length
'CLASSES:'              - Highway classification sets
'U EF'                  - Urban Expessways & Freeways
'R'                     - all Rural highways
'VOLUMES:'              - Average Daily Traffic (ADT) ranges
0 5000
5001 10000
10001 999999
```

THAS260 Structure Chart



* Structure charts for these subroutines are shown (once only) elsewhere on the page.

Utility: ABT

Rate Table Report Layout (SECTIONS)

```
         1         2         3         4         5         6         7         8         9        10        11        12
1234567890    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0
```

***** A C C I D E N T R A T E T A B L E *****
Version 4.0

Data Selection Dates: yr/mo/da - yr/mo/da Months: mo - mo Hours: hr - hr
Report Date: yr/mo/da hr:mn Page ZZ9

Calculation Type: SECTIONS Accident Rate Units: Accidents per million vehicle-km.
DIVIDED -- A:ZZ9.9 A:ZZ9.9 A:ZZ9.9 A:ZZ9.9 km. Fatal Accident Rate Units: Accidents per 100-million vehicle-km.

<---------- User-defined Title ---------->

Traffic Volume Ranges (ADT)	----- Highway Classification Sets -----								
	1 cl.set	2 cl.set	3 cl.set	4 cl.set	5 cl.set	6 cl.set	7 cl.set	8 cl.set	9 cl.set
ADT: ZZZZZZ9 -ZZZZZZ9									
Wtd. Avg. Acc. Rate	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999
Wtd. Avg. Fatal Acc. Rate	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999
Road-Weighted Acc. Rate	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999
Road-Wtd. Fatal Acc. Rate	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999
Mean Acc. Rate	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999
Standard Deviation (Sigma)	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999	ZZ9.999
N (# Sections)	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9
% Fatal Accidents	ZZ9.99	ZZ9.99	ZZ9.99	ZZ9.99	ZZ9.99	ZZ9.99	ZZ9.99	ZZ9.99	ZZ9.99
% Injury Accidents	ZZ9.99	ZZ9.99	ZZ9.99	ZZ9.99	ZZ9.99	ZZ9.99	ZZ9.99	ZZ9.99	ZZ9.99
% PDO Accidents	ZZ9.99	ZZ9.99	ZZ9.99	ZZ9.99	ZZ9.99	ZZ9.99	ZZ9.99	ZZ9.99	ZZ9.99
# Fatal Accidents	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9
# Injury Accidents	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9
# PDO Accidents	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9
Total Accidents	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9	ZZZZZZ9
Total Length (Km)	ZZZZ9.9	ZZZZ9.9	ZZZZ9.9	ZZZZ9.9	ZZZZ9.9	ZZZZ9.9	ZZZZ9.9	ZZZZ9.9	ZZZZ9.9


```
         1         2         3         4         5         6         7         8         9        10        11        12
1234567890    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0
```


If a SECTIONS-type Rate Table does not have its sections divided into smaller pieces, the heading shows "UNDIVIDED" below the Calculation Type, instead of "DIVIDED" and up to 4 hwy. classes and section lengths.

Rate Table Report Layout (LOCATIONS or LANDMARKS)

```
         1         2         3         4         5         6         7         8         9        10        11        12
1234567890    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0


                                  ***** A C C I D E N T   R A T E   T A B L E *****
                                                  Version  4.0

Data Selection Dates:  yr/mo/da - yr/mo/da     Months:  mo - mo     Hours:  hr - hr
Report Date:  yr/mo/da   hr:mn                                                                                  Page ZZ9

Calculation Type:  LOCATIONS AT LANDMARKS                    Accident Rate Units:  Accidents per     million vehicles
  Landmark Types:  ZZ ZZ ZZ ZZ ZZ ZZ ZZ ZZ ZZ ZZ       Fatal Accident Rate Units:  Accidents per 100-million vehicles

                   <----------------------------- User-defined Title ----------------------------->

                                ----- Highway Classification Sets -----
          Traffic Volume        |
           Ranges (ADT)         |   1    |   2    |   3    |   4    |   5    |   6    |   7    |   8    |   9    |
                                | cl.set | cl.set | cl.set | cl.set | cl.set | cl.set | cl.set | cl.set | cl.set |
------------------------------+--------+--------+--------+--------+--------+--------+--------+--------+--------+
|      ADT: ZZZZZZ9 -ZZZZZZ9 ||        |        |        |        |        |        |        |        |        |
|        Wtd. Avg. Acc. Rate ||ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |
|                            ||        |        |        |        |        |        |        |        |        |
|  Wtd. Avg. Fatal Acc. Rate ||ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |
|                            ||        |        |        |        |        |        |        |        |        |
|    Road-Weighted Acc. Rate ||ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |
|                            ||        |        |        |        |        |        |        |        |        |
|  Road-Wtd. Fatal Acc. Rate ||ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |
|                            ||        |        |        |        |        |        |        |        |        |
|             Mean Acc. Rate ||ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |
| Standard Deviation (Sigma) ||ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |ZZ9.999 |
|            N (# Landmarks) ||ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |
|                            ||        |        |        |        |        |        |        |        |        |
|          % Fatal Accidents || ZZ9.99 | ZZ9.99 | ZZ9.99 | ZZ9.99 | ZZ9.99 | ZZ9.99 | ZZ9.99 | ZZ9.99 | ZZ9.99 |
|         % Injury Accidents || ZZ9.99 | ZZ9.99 | ZZ9.99 | ZZ9.99 | ZZ9.99 | ZZ9.99 | ZZ9.99 | ZZ9.99 | ZZ9.99 |
|            % PDO Accidents || ZZ9.99 | ZZ9.99 | ZZ9.99 | ZZ9.99 | ZZ9.99 | ZZ9.99 | ZZ9.99 | ZZ9.99 | ZZ9.99 |
|                            ||        |        |        |        |        |        |        |        |        |
|          # Fatal Accidents ||ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |
|         # Injury Accidents ||ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |
|            # PDO Accidents ||ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |
|                            ||        |        |        |        |        |        |        |        |        |
|            Total Accidents ||ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |ZZZZZZ9 |
|                            ||        |        |        |        |        |        |        |        |        |
|          Total Length (Km) ||ZZZZ9.9 |ZZZZ9.9 |ZZZZ9.9 |ZZZZ9.9 |ZZZZ9.9 |ZZZZ9.9 |ZZZZ9.9 |ZZZZ9.9 |ZZZZ9.9 |
|                            ||        |        |        |        |        |        |        |        |        |
------------------------------+--------+--------+--------+--------+--------+--------+--------+--------+--------+

         1         2         3         4         5         6         7         8         9        10        11        12
1234567890    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0    5    0
```

For a LOCATION-type table, the Calculation Type is given as 'LOCATIONS WITH ACCIDENTS'; no landmark types are shown, and the third statistics line of each cell is labelled 'N (# Locations)' instead of 'N (# Landmarks)'. Otherwise the layout is as shown above.

8.2.14.4 Equal-Volume Sections

*** NOT IMPLEMENTED ***

What follows is a brief description of a modification to the Rate Table that was originally intended to be part of version 4.0, that is, ensuring that each section has a constant volume (SECTION-type tables). After discussions with Highway Safety, it was decided not to make this modification.

The objective would be to produce a Highway Sections Table which starts a new section when either the highway classification or traffic volume changes.

Determining the places where traffic volume changes has a complication, in that traffic-volume counters are often moved from one year to another. Therefore, Highway Safety may specify different counters for a section of road in different years. Thus, a segment may be split into equal-volume sections at different points each year.

The most reasonable interpretation of "equal-volume sections" for the Rate Table is probably to split a segment wherever the volume changes in *any* year (ignoring years that are outside the date-selection range of the accident data). This could cause many small sections with different volumes to be created within a single segment, and almost all of them would fall in the same traffic-volume range of the Rate Table. In this case, the data would become difficult and confusing to interpret; that is why this modification has not been done at this time.

Algorithm

Below is an algorithm showing how segments could be split into equal-volume segments. It requires a new routine, THASFBP, and extensive changes and rearrangements of existing code.

Overall algorithm:

```
THASLSC:  Load the SEGCLASS table (SCLTAB).
THASLSP:  Load the Search Path (PATH).
THASRNS:    Modify the Search Path table (PATH) so nodes are not
            included as part of section-type entries.
THASSPS (Creates the Highway Sections Table):
      FOR every entry in the Search Path:
            IF it's a node-type entry:
                  THASANS: Add a node-section to the HSECTAB.
            ELSE IF it's a segment-type entry:
                  THASFBP (new):    Create the breakpoints-array for
                                    this segment.
                  Set B to PATH.STARTKM.
                  For each entry in the breakpoints-array:
                    Set E to the breakpoint-KMMARK.
                    Add the segment-section B through E to the HSECTAB.
IF the user wants small sections, then:
      THASDSS (modified): Divide the HSECTAB into smaller pieces.
```

Description of new routine THASFBP (Find_BreakPoints):

Given a segment number and a start- and end-KMMARK from the Search Path, this routine returns an array of KMMARKs that are the endpoints of sections of constant volume and highway-class. Some provision will need to be made for including sections with a missing volume or class. This is an exceedingly complex question, since a section may have a missing traffic volume for some years but not for others; therefore, the possibility of missing class/volume has not been accounted for in the algorithm given.

(THASFBP could optionally be designed to ignore volume breakpoints where the year's average ADT has not changed sufficiently to put a section in a different Rate Table cell.)

Algorithm for THASFBP:

```
Open SEGVOL (the selected segment-volume file).
Read through SEGVOL until the first record for the segment is found.
Store the end-KMMARK in the breakpoint-array (only entry so far).
Do until the last record for the segment is found:
      If the record is entirely outside the segment KMMARK-range, then
            ignore it.
      Set K to the section-endpoint on this record.
      If K is not yet in the breakpoint-array and K < segment end-KM,
            then insert K in the breakpoint-array (so the array remains
            sorted by increasing KMMARK).
Close SEGVOL.
Open SEGCLASS (the segment-classification file).
Read through SEGCLASS until the first record for the segment is found.
Do until the last record for the segment is found:
      Set K to the section-endpoint on this record.
      If K is not yet in the breakpoint-array and K < segment end-KM,
            the insert K in the array.
Close SEGCLASS.
```

8.2.15 Program THAS270 - Calculate Average Accident Type Ratios

Purpose: Calculate average accident type ratios, for each reference group and accident type. The ratios are written to a report, and optionally to a disk file.

Input:

SYSIN	- control flags to turn the following features on/off: - produce ratio report - write average ratios to the RATIOS file - define location reference groups, each with a Highway Classification Set, and up to 6 landmark type codes - define section reference groups, each with a Highway Classification Set - see Notes, below for an example reference group definition.
INDATA	- input accident data file
INDESC	- description of INDATA
LANDMRK	- LKI Landmark file.
ACCTYP	- THASP.TABLE(ACCTYPES) - copied to the report as a key the accident types.

Output:

LRATIOS	- disk file of average ratios for Locations
SRATIOS	- disk file of average ratios for Sections
SYSPRINT	- messages, summary etc.
REPFILE	- average accident type ratio report
SDATA	- accidents which are in any Location reference group.
SDESC	- description file for SDATA
XDATA	- accidents NOT in any Location group, and IN any Section reference group.
XDESC	- description file for XDATA

Notes:

Locations in this program are identified only by landmarks on the Landmark file. (Not by the location type field in the accident records.)

This program was written and documented with intersections in mind, but landmark codes could be specified to define railway crossings,bridges, and private drives, as the three location types if desired.

Reference groups:

Each reference group is defined with a name, a Highway Classification Set, and up to 6 landmark codes. If the landmark code string is non-blank, then it is a Location reference group. If the landmark code string is blank, then it is a Section reference group. Landmark codes must be blank-separated. See the H.A.S. User's manual for a definition of Highway Classification Sets.

The following example defines three location reference groups followed by two section reference groups:

```
'LOCATIONS'
'Freeway Intersections'  '* F'  'A1 A2 A3 A4 A5'
'Urban Signalized'       'U'    '#1'
'Rural Uncontrolled'     'R'    'A2 A4'
'END'
'SECTIONS'
'Urban Conventional'     'U C'
'Express and Freeways'   '* EF'
'END'
```

Reference Group Names are used in the report, and are written to the Average Accident Type Ratios files.

The location reference group definitions must come first in the SYSIN file.

The following rules are applied to determine which reference groups an accident may belong:

- An accident is first tested to see if it is in a Location group
- An accident is eligible for entry in a Section group if and only if the accident is NOT in a Location group.
- An accident may be in:
    - one or more Location groups, OR
    - one or more Section groups, OR
    - in no group at all.

Accident Types

The following accident types are used. Note that it is possible for an accident to be of none of these accident types, or be of more than one accident type.

1. right angle
2. left turn opposing
3. straight ahead rear end
4. left turn rear end
5. right turn
6. sideswipe
7. head on
8. off road
9. fixed object
10. parked
11. general rear end
12. pedestrian
13. animal

Accident type definitions are coded in procedure THASSAT.

Description

- define 3-dimensional arrays: P(2,10,13)
  X(2,10,13)
  using indeces: K, G, T

  where: K is 1 for locations, 2 for sections,
  G is the reference group number,
  T is the accident type number

FOR each accident:
- determine the accident type, as described above, setting T

- look up the landmark type(s) for the accident location on the landmark file.
- set K = 1
- for each Location reference group, if the highway class of the accident is included in the Class Set of the group, and if any of the landmark types from the landmark file match any of those of the group, then increment X(K,G,T), where G is the reference group number.

- if the accident was not counted in any of the location reference groups, THEN:
  - set K=2
  - for each Section reference group, if the highway class of the accident is included in the Class Set of the group, increment X(K,G,T), where G is the reference group number.

- If the accident was in NO reference groups, location or section, set K= 0

- IF K = 1
  - if a Location subset has been requested, write this record to the SDATA file

- IF K = 2
  - if a Section subset has been requested, write this record to the XDATA file.

ENDFOR

FOR each value of K (1 and 2):
FOR each reference group (G = 1 to ?)
- sum the row (T = 1 to 13) to get the total number of accidents in the reference group
- divide each element of the row by the sum, to get the average accident type ratio, and store it in P(K,G,T)

ENDFOR
ENDFOR

If requested, write the ratios out to a report and to files LRATIOS and SRATIOS.

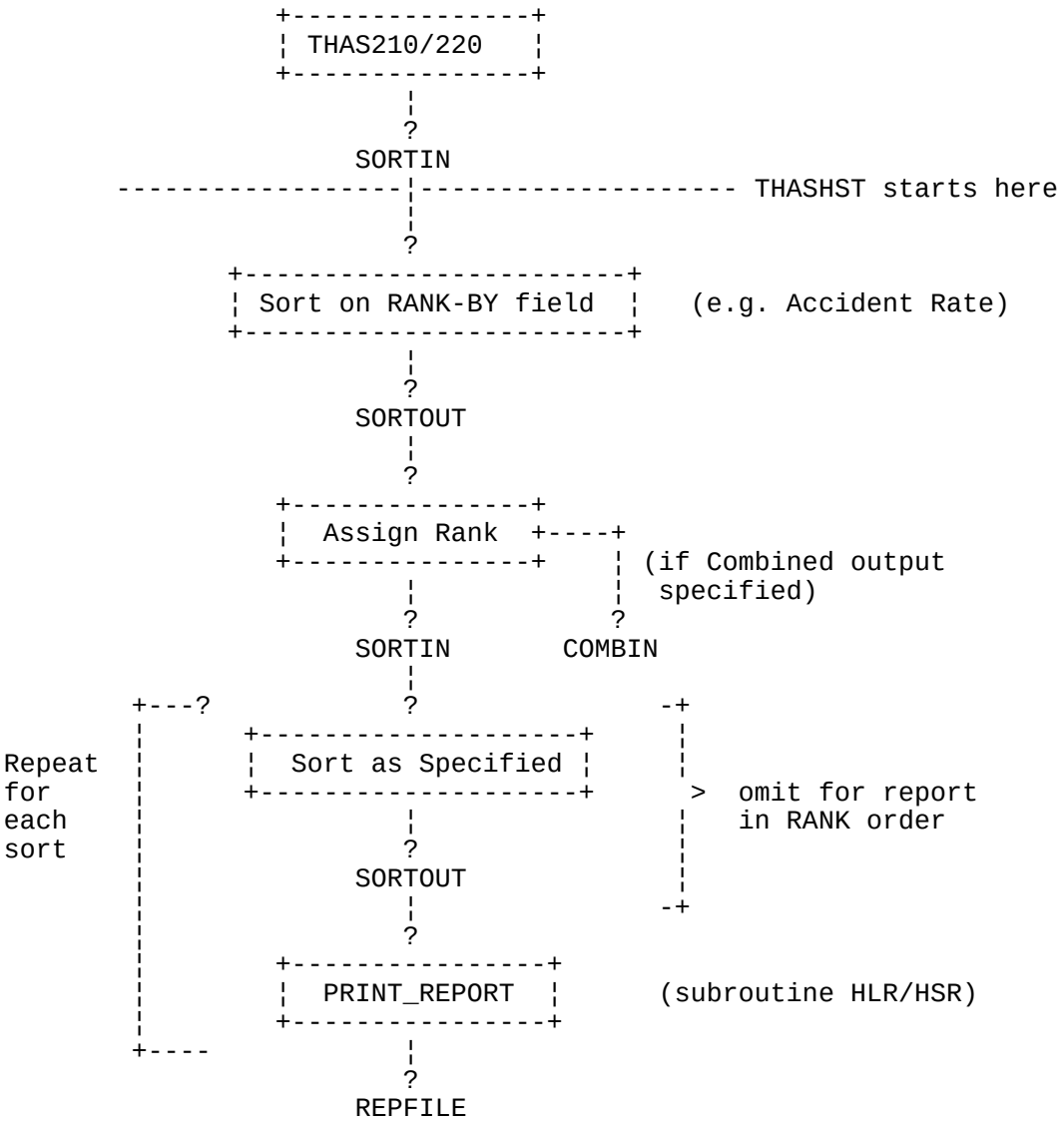
8.2.16 Program THAS710 - Create Victim File

```
Purpose:    to produce an accident data file with one record per victim, so
            that a subsequent SAS program can easily analyse victim
            information.

Input:      INDATA        -   input accident data file
            INDESC        -   description of INDATA

Output:     VICTIM        -   one 151 byte accident record per victim
            VICDESC       -   description file
            SYSPRINT      -   messages and printed report

Description:

For each victim in each input accident record, produces a 151 byte victim
record containing the first 136 bytes of the accident record followed by the
15 bytes of victim information.

If there is no victim information in an accident record, no victim records
are produced.

See option V on panel THASPSEL.
See JCL skeleton THASPVIC.
```

8.3 Subroutine THASHST - Sorting Haz. Loc./Sec. Report Records

This following chart illustrates how subroutine THASHST produces Hazardous Location and Section reports in one or more requested orders.

Program functions are in boxes, and DD names are not.



8.4 Notes and Enhancement Ideas (for Data Retrieval)

Most of the following are things which were left during development as things to do "if there was time"!

THAS200

1. Count and report the number of duplicate records/accidents selected when node selection = M.

THAS210 & THAS220

1. Catch Hazardous Locations at Landmarks when radius > 0 in cases where there are no accidents right at the location, but plenty within the radius.

2. The sort field information is presently hard coded in the programs (field name and position, etc.) The field position and length info could be added to the DEF210 and DEF220 files in THASP.DEFAULTS file, be transferred to the SYSIN file of the JCL, then be read in by the program, instead.

THAS230

1. An option could be added to the REXX program and the JCL to make SMRYFIL a permanent file when desired. An option could also be added to print SMRYFIL as an extra report.

2. Obsolete-location accidents of each type at each location could be counted and added to the summary file, and lines of the summary file containing obsolete-location accidents could be printed out at the end of the Histogram.

ISPF Dialog

1. The From-To Highway & Segment verification sections of program THAS030 could be extracted into a separate program and run in foreground by the ISPF Dialog, so that errors would be detected at data entry time.

Node Analysis Program

A Node Analysis Program could be written to access the PDS Master directly, and search a specified distance up EVERY segment connecting at each node. This would make up for the limitation of the Hazardous Locations program that it can only search along the Search Path at nodes.

9 Utility Sub-System

9.1 Introduction

The Utility system consists of the jobs run from the Utilities menu and its sub-menus.

The following jobs can be run from the Utilities menus:

Job 752	Creates a CSVfile of Traffic volumes for specified segments.
Job PDSM	Creates new PDS Master Files from the Tape Master File.
Job PDSI	Implements the new PDS Master Files created by job PDSM.
Job 720	Creates a new generation of the master file THASP.PROV.VALLOC.CURRENT, making specified changes to specified accident records.
Job 760	Creates the Station Volumes file from the traffic volume files obtained from TIMS.
Job 761	Creates the H.A.S. traffic volume files from the Station Volumes file, as specified in the Subsegment, Intersection and Node Map files.
Job 770	Prepares the SEGCLASS file for use by adding sequence numbers and sorting it by HSNFIL order and KMMARK.
Job 775	Creates file THASP.SEGDIST.SORTED from file THASP.SEGDIST.

See the beginning of the Update Job Flowchart section for notes on interpreting the flowcharts.

9.2 PDS Master Files Generation

9.2.1 Job PDSM - Create the new PDS Master Files

1. Program THAS132 reformats the SEGMENT file into the SHNFIL file.

2. Program THAS134 inverts the segment/node information in the SEGMENT file to create NODESEG. File NODESEG allows people and programs to look up a node name and see what segments connect to it.

3. Program THAS101 selects the desired year range of data from the master accident file.

4. Program THAS105 reads the master file of valid-location accidents, and converts pre-1990 location codes to location codes according to the 1995 LKI.

5. Program THAS152 does a fix on segment 0720 location codes because the Burnaby police were using the wrong LKI in the 1990-1995 period.

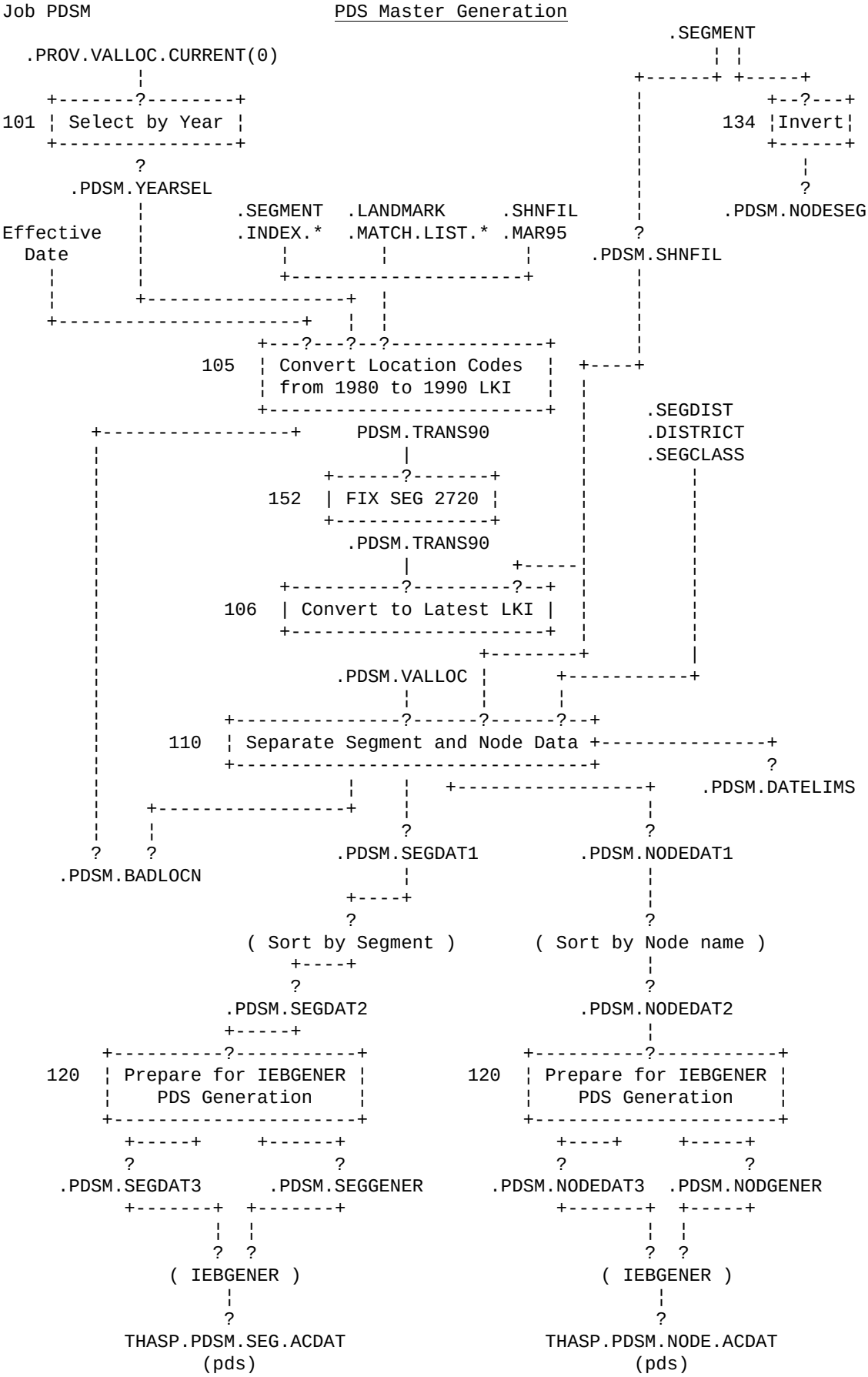
6. Program THAS106 does converts the location codes to the current LKI.

7. Program THAS110: the location codes are compared to information on the SHNFIL, and the accidents are separated into accidents at Nodes, and accidents not at Nodes. Accidents with location codes inconsistent with the segment data on file SHNFIL are separated - they do not go into the PDS Master. LOCN_ID, Region, District and Day-of-Week are inserted into the accident records at this point.

8. Node data is put into the Node PDS, and segment data is put into the Segment PDS, with one member per node or segment. For both nodes and segments, program THAS120 creates a file of specially prefixed accident records and a matching set of IEBGENER control statements, then IEBGENER does the PDS creation.

Note that job PDSM names all its new and intermediate files with PDSM as the second index level. No production files are replaced. This allows the job to be run and checked thoroughly without interrupting production use of the H.A.S. If anything goes wrong, all intermediate files are cataloged, for analysis or re-starts.

Job PDSI implements the new files, and deletes the intermediate files.


Job PDSM
PDS Master Generation
.PROV.VALLOC.CURRENT(0)
101 Select by Year
.PDSM.YEARSEL
Effective Date
.SEGMENT
.INDEX.*
.LANDMARK
.MATCH.LIST.*
.SHNFIL
.MAR95
.SEGMENT
134 Invert
.PDSM.NODESEG
.PDSM.SHNFIL
105 Convert Location Codes from 1980 to 1990 LKI
PDSM.TRANS90
.SEGDIST
.DISTRICT
.SEGCLASS
152 FIX SEG 2720
.PDSM.TRANS90
106 Convert to Latest LKI
.PDSM.VALLOC
110 Separate Segment and Node Data
.PDSM.DATELIMS
.PDSM.BADLOCN
.PDSM.SEGDAT1
.PDSM.NODEDAT1
(Sort by Segment)
(Sort by Node name)
.PDSM.SEGDAT2
.PDSM.NODEDAT2
120 Prepare for IEBGENER PDS Generation
120 Prepare for IEBGENER PDS Generation
.PDSM.SEGDAT3
.PDSM.SEGGENER
.PDSM.NODEDAT3
.PDSM.NODGENER
(IEBGENER)
(IEBGENER)
THASP.PDSM.SEG.ACDAT
(pds)
THASP.PDSM.NODE.ACDAT
(pds)

9.2.2 Job PDSI - Implement the new PDS Master Files

Job PDSI implements the new files and deletes the intermediate files created by job PDSM.

The following files are deleted using IDCAMS:
(The THAS? first index level is omitted)

```
PDSM.YEARSEL
PDSM.VALLOC
PDSM.SEGDAT1
PDSM.NODEDAT1
PDSM.SEGDAT2
PDSM.NODEDAT2
PDSM.SEGDAT3
PDSM.SEGGENER
PDSM.NODEDAT3
PDSM.NODGENER
```

The following files are renamed to remove the PDSM second index level, using TSO RENAME statements in a REXX EXEC started in the batch job:

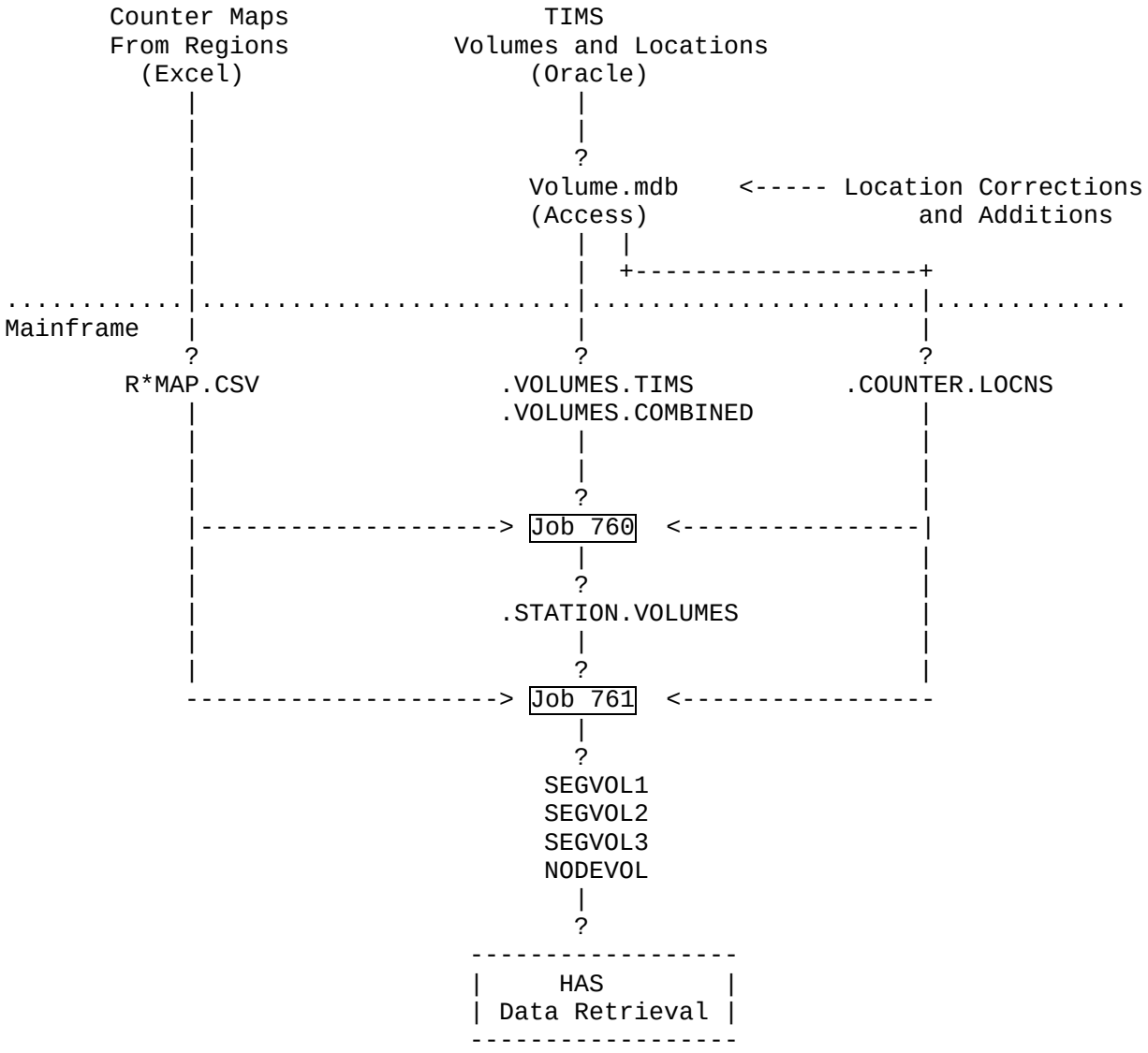
```
PDSM.SEG.ACDAT
PDSM.NODE.ACDAT
PDSM.SHNFIL
PDSM.NODESEG
PDSM.DATELIMS
PDSM.BADLOCN
```

9.2.3 Job PDSA - Archive PDS-Master

The Archive PDS-Master files are created using JCL in THASP.JCLIB(ARCPDSM). DARCPDSM should be run afterwards to clean up intermediate files. Development versions (THASD) of this JCL are in THASD.UTILJCL.

9.3 Traffic Volume Files Generation

9.3.1 Overview



9.3.2 Counter Maps - Summary

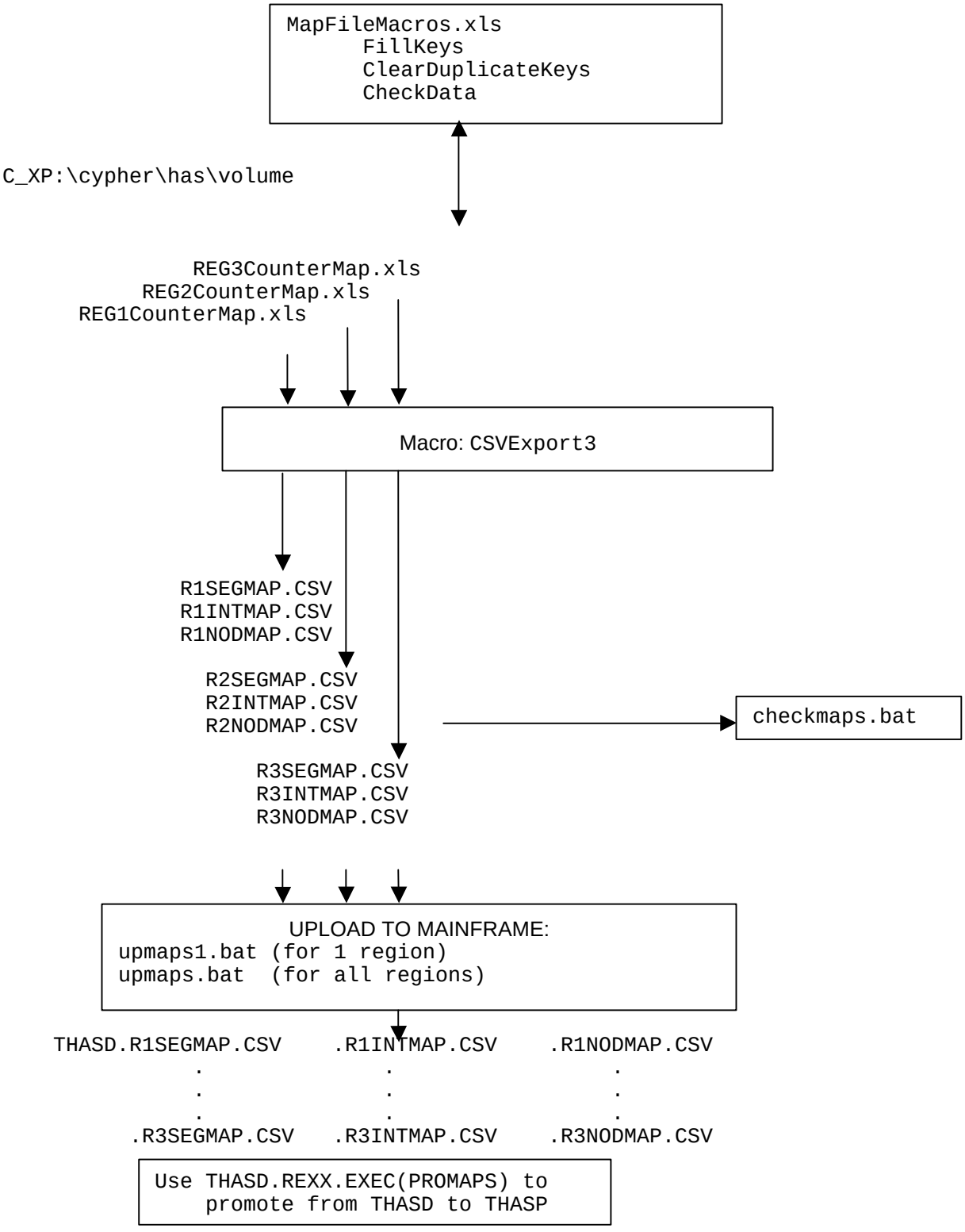
See the **Counter Mapping** section of the User manual for an introduction to counter maps.

The counter maps are maintained in Excel files (workbooks), one per region. Each file contains 3 worksheets: one each for Subsegments, Intersections and Nodes. The counter maps are maintained at Cypher Consulting, where they are kept on a Windows workstation. Using Excel Macros kept in an Excel file called MapFileMacros.xls, the map worksheets are checked, key fields filled in, and exported to CSV files.
The CSV files are uploaded to the mainframe. Volume utility Job 760 combines and sorts the map files into three files: one for Subseg, Intersections and Nodes. See the charts on the next pages for file names.

A copy of MapFileMacros.xls is kept on the public drive, in folder:
P:\HQ\ENG\safety\has\Volume\.

The Regional Excel counter map files could be reconstructed if necessary from the RntypMAP.CSV files which are on the mainframe.

9.3.3 Counter-Map Processing and Upload Chart



9.3.4 Obtaining Traffic Volumes from TIMS

Volumes were last obtained from the TIMS database in July 2002 by contacting David.Godfrey@gems6.gov.bc.ca, and asking him to perform the following queries on the TIMS Oracle database:

```
QUERY 1: From the table(s) which contains Station Locations:

      TM Point
      Road Name
      Location (description)
      District Number (or Name)
      Count Location
      LKI Segment
      LKI Driven Distance

QUERY 2: From the Count table(s)

      TM Point
      Year
      AADT
      JAN ... DEC MADTs.
```

The results were delivered via e-mail in Excel files.

9.3.5 The VOLUME MS-Access Database

The traffic volumes obtained from the TIMS, and the counter locations (from TIMS and other sources) are maintained in an MS-Access database on the Cypher Consulting workstation.

A copy of the volume database can be found in folder:
\\P:\HQ\ENG\safety\has\Volume,
in file **VolumeDatabase.zip** and/or **Volume.mdb**

The database contains the following tables:

Counter_Locations

- fields: CountID, Region, District, RoadName, Segment, KM and Description.
- contains all the counters, with and without LKI locations (only those with LKI locations are exported for the COUNTER_LOCNS file.)
- much of this data came from TIMS, but many missing LKI locations were determined and inserted afterwards.
- the LKI locations will have to be converted any time a Segment is significantly modified (re-measured).
- Count ID's end in N, S, E or W if the count is for one direction of a 2-way road.

Counter_Locations_Combined

- same fields as in the Counter_Locations table
- contains ID's created from the ID's of uni-directional counts in the Counter_Locations table, for all stations on 2-way LKI segments.
- Count ID's end in C.
- This table is generated (and can be re-generated) using the Combine_Counter_Locations function in the Combine module in the database.

Volumes_TIMS

- each record contains the count ID, year, AADT and 12 MADTs.
- this is the data as obtained from TIMS
- Count ID's end in N, S, E or W if the count is for one direction of a 2-way road.

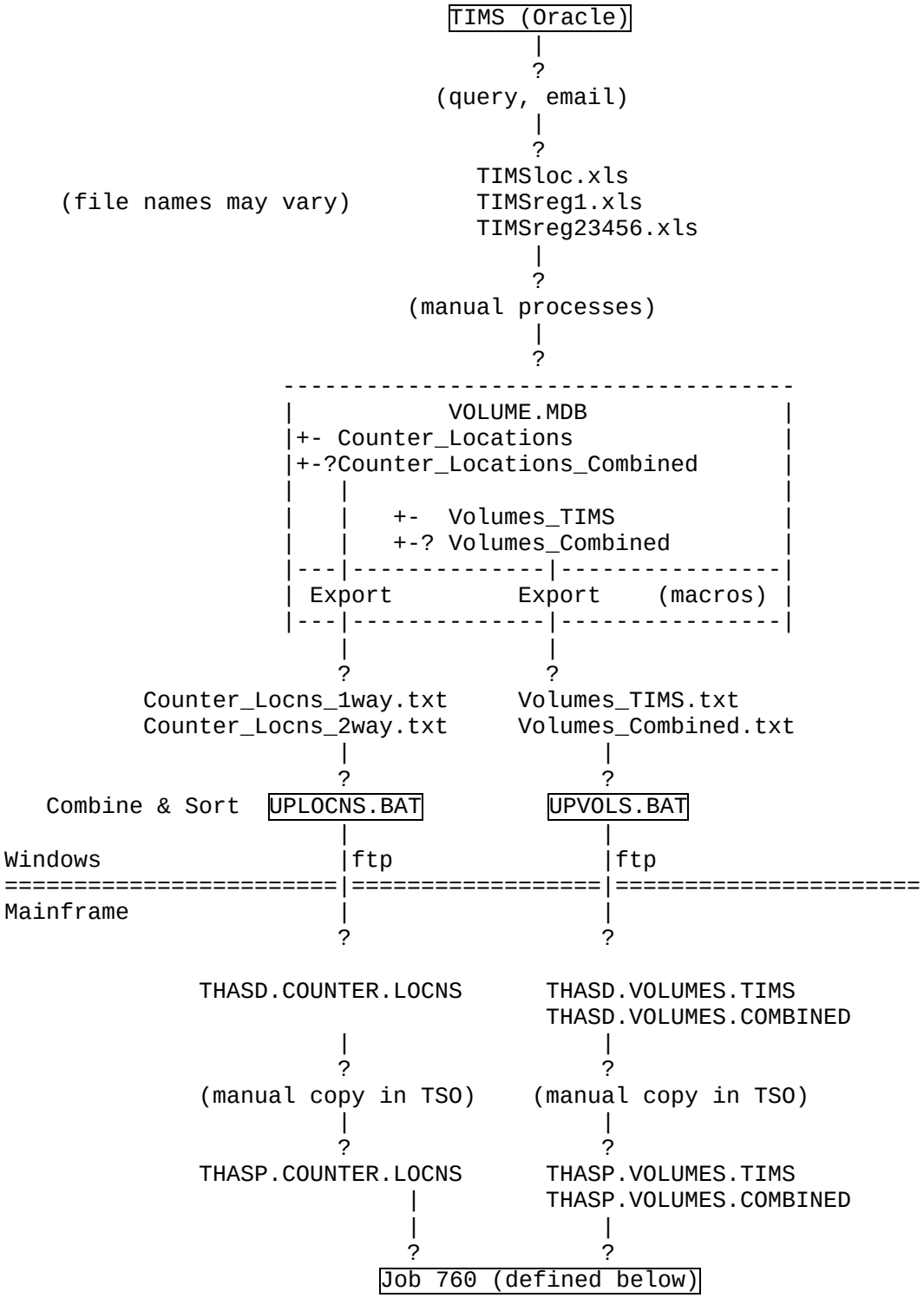
Volumes_Combined

- same fields as in the Volumes_TIMS table
- contains counts combined from the uni-directional counts in the Volumes_TIMS table, for all stations on 2-way LKI segments.
- Count ID's end in C.
- This table is generated (and can be re-generated) using the Combine_Volumes function in the Combine module in the database.

Exporting Data for HAS

In the database, use macros **ExportCounter_LocationsTable** and **ExportVolumesTable** to export the data into formatted text files. These macros use functions in the **Export** VBA module.

9.3.6 Volume Preparation and Upload Chart



9.3.7 Job 760 - Create the Station Volumes File

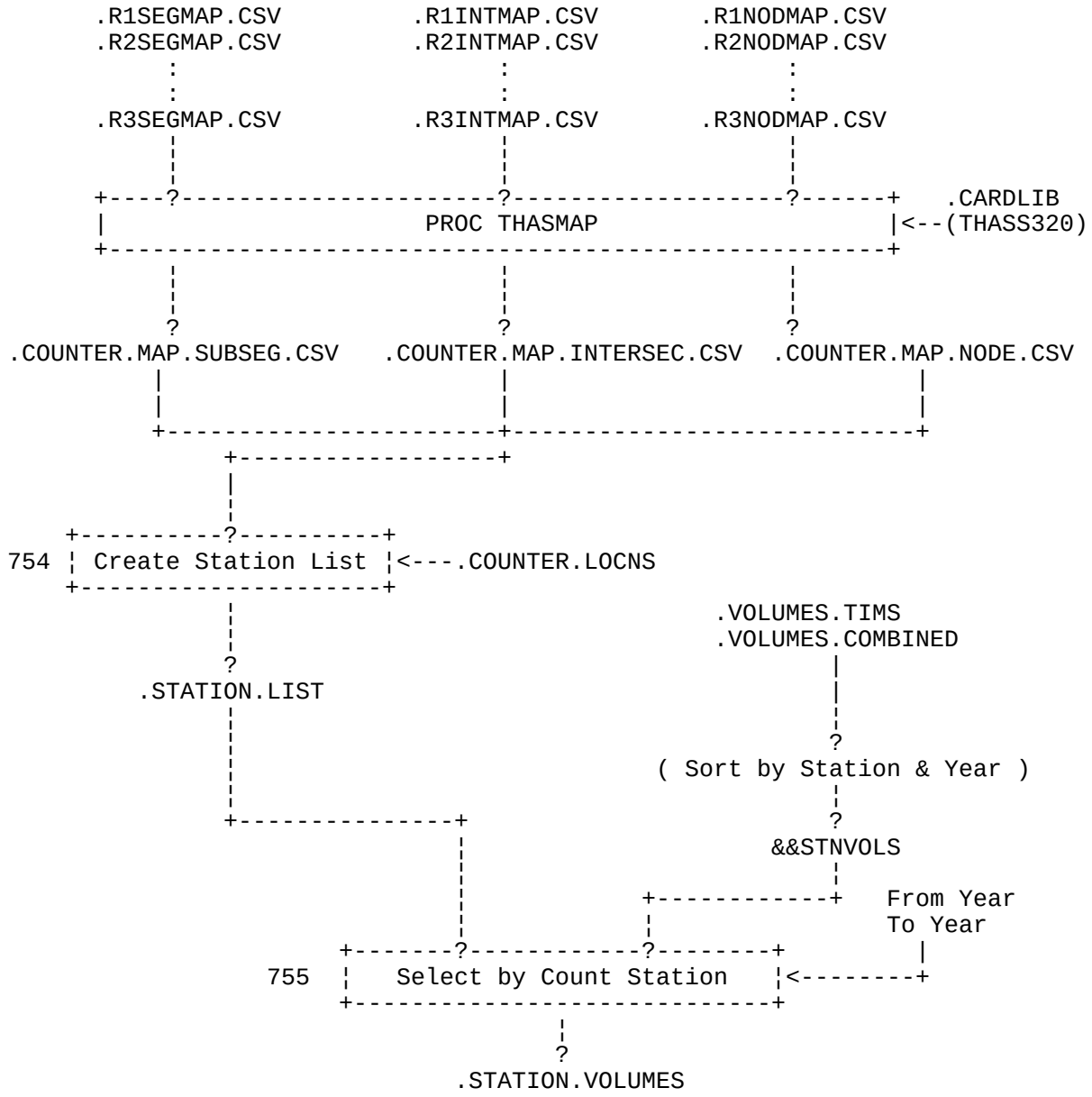
This job extracts from the TIMS data, the data for the years and count stations needed for the H.A.S.

1. Proc THASMAP combines and sorts the counter map files into one file per type (SubSeg, Intersec, Node)

2. Program THAS754 reads all the H.A.S. files in which count stations are specified, and creates a sorted unique list of all the count stations whose data are required by the H.A.S.
   Note: file THASP.COUNTER.LOCNS contains only counters on the LKI.

3. Program THAS755 extracts from the VOLUMES.TIMS and VOLUMES.COMBINED files the traffic volume data of the count stations required by the H.A.S., creating the Station Volumes file.

Job 760 - Create Station Volumes File



9.3.8 Job 761 - Create HAS Production Traffic Volume Files

1. Program THAS760 uses the Station Volumes file and the Counter Locations file to determine average traffic volumes for each segment containing traffic volume stations, as well as for each segment with stations on neighbouring segments.

   This is an automatic, but crude process, which produces traffic volumes which are the same for each entire segment.

   (This is the only traffic volume file which was used in the H.A.S. prior to April 1995.)

2. Sub-Segment MAP Processing:
   - Program THAS762 implements the instructions coded in the COUNTER.MAP.SUBSEG.CSV file, calculating traffic volumes for all specified sub-segments.
   - if a map results in no data for a year, no output is produced for that year - i.e. it will not replace SEGVOL1 data with zeros for an entire year.
   - Program THAS766 takes the sub-segment volumes, and fills in all the blanks with data from file SEGVOL1, produced in the previous step. I.e. the crude data is refined with the sub-segment data.

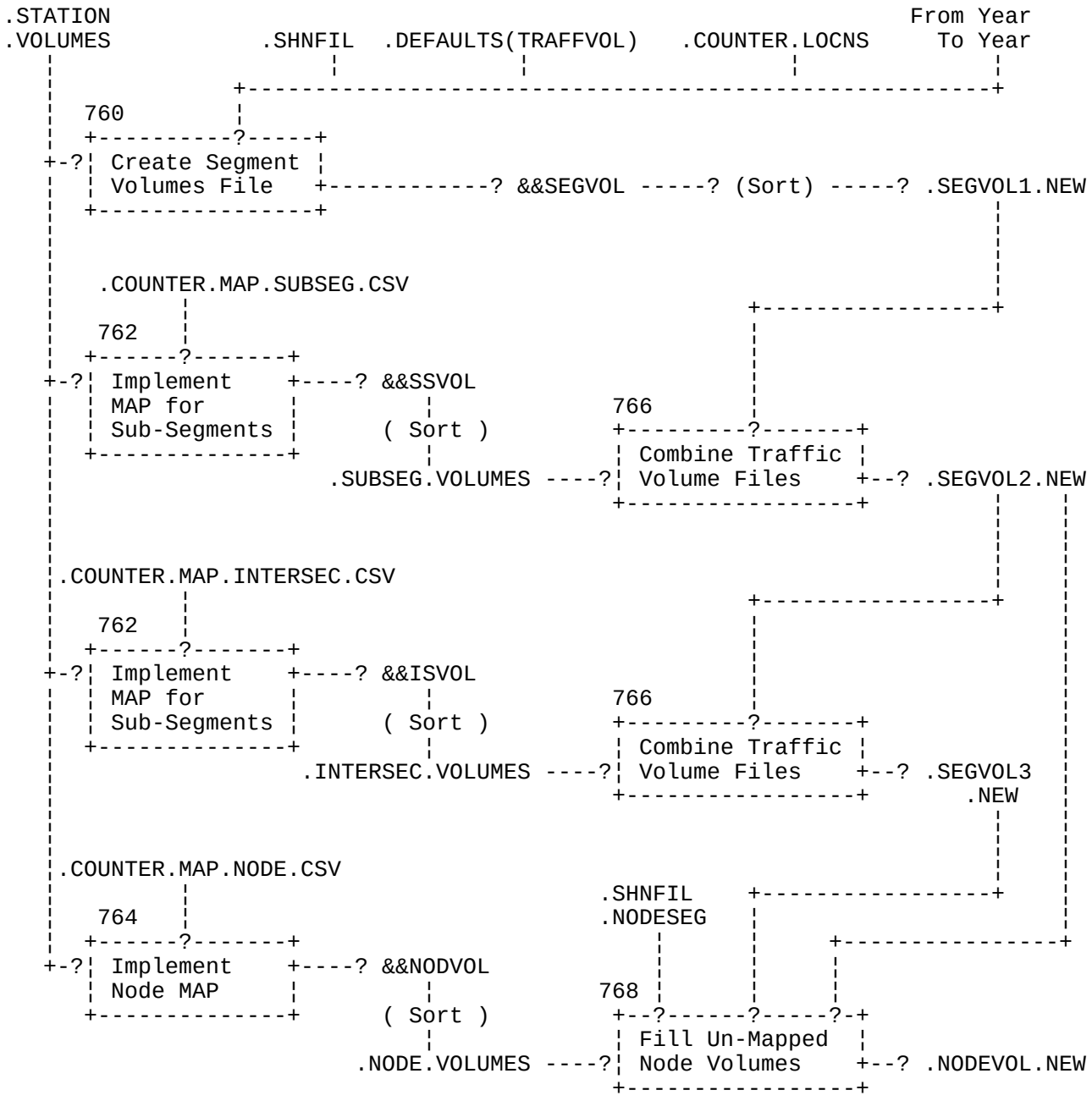
3. Intersection MAP Processing:
   - This process is identical to the sub-segment process: an intersection is logically just a short sub-segment.
   - Program THAS766 further refines the SEGVOL2 file with the intersection data.

4. Node MAP Processing:
   - Program THAS764 implements the instructions coded in the COUNTER.MAP.NODE.CSV file, calculating traffic volumes for all specified nodes.
   - Program THAS768 calculates traffic volumes for all nodes not specified in the MAP file, by averaging adjacent sub-segment volumes on the segments named in the node name.

Note: All the new files created are named with '.NEW' on the end. These files are renamed (put into production) in foreground TSO, as an option on the Traffic Volume Utilities panel. (See REXX program THASP.ISPF.ISPCLIB(THASUVOL).)

If, at any stage, data is not available for all years requested, the first year of data may be duplicated back one year, the last year of data may be duplicated forward one year, and missing years will be filled by interpolation. (See PL/I routine THASSTV)

Job 761 - Create H.A.S. Traffic Volume Files



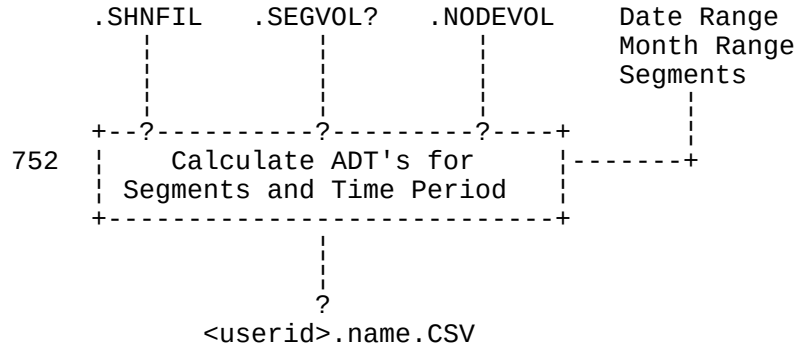
Note: all sorts are by Segment, Year and (where applicable), Start_Km

9.3.9 Job 752 - Create Traffic Volume CSV File

Program THAS752:
- Reads a date range and a month range from SYSIN.
- Reads a list of segments from SYSIN (or 'ALL')
- For each segment, the segvol records are scanned for the years in the date range, and the start and end points of all volume sub-segments are recorded.
- Then THASVSS is called for each of those sub-segments to get a volume.
- The node volumes are included.

The output CSV file has the following fields:

Segment
Start Km
End Km
ADT
NodeName



Notes:
- the user specifies which SEGVOL file is used (1, 2 or 3)
- the user specifies the name for the output file.

9.4 Miscellaneous Utilities

9.4.1 Job 720 - Modify the (Tape) Master File.

This job is used to correct data on the master file. After it has been run, job PDSM must be run to make the changed data available to the Data Retrieval System.

1. The user identifies accident records to modify by specifying the Accident Case Number, and by specifying either a particular page number, or ALL pages of the accident.

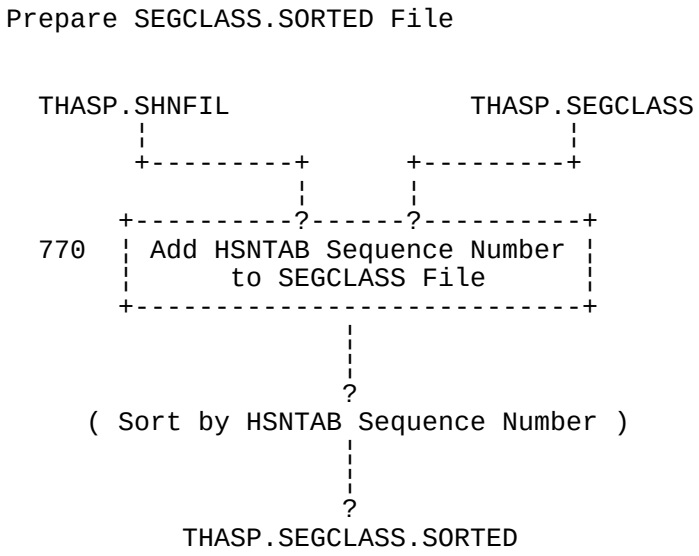
2. Fields to modify are identified by field name. Program THAS720 looks up the location of the field using subroutine THASFLD.

3. A new generation of the (Tape) Master File is created.

9.4.2 Job 770 - Prepare file SEGCLASS.SORTED

1. For each record of the SEGCLASS file, program THAS770 determines the index of the segment within the Highway-Segment-Node table and adds this index to the record as the "HSNTAB Sequence Number."

2. The SEGCLASS file is sorted by HSNTAB Sequence Number and KMMARK, thus putting it into Search Path order.

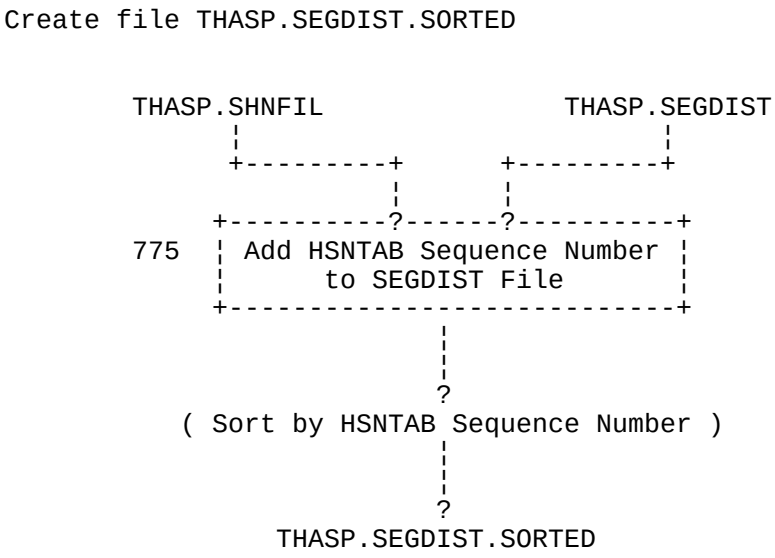


9.4.3 Job 775 - Prepare file SEGDIST.SORTED

1. For each record of the SEGDIST file, program THAS775 determines the index of the segment within the Highway-Segment-Node table and adds this index to the record as the "HSNTAB Sequence Number."

2. The Offset is added to the Start KMMARK and End KMMARK, then set to zero.

3. The SEGCLASS file is sorted by HSNTAB Sequence Number and KMMARK, thus putting it into Search Path order.



9.5 Program Descriptions

Summary

Main Routines

THAS105 - converts location codes to the 1995 LKI.
THAS106 - updated version of THAS105: converts from 1995 to current LKI
THAS110 - separates segment and node accidents.
THAS120 - prepares data file and IEBGENER control statements for PDS generation.
THAS132 - combines LKI Highway and Segment info with Node, continuity, and SEGWAY information to create SHNFIL.
THAS134 - inverts the Segment-Node file to create the Node-Segment file.
THAS700 - creates old-new landmarks file.
THAS720 - copies an accident file, making specified data changes.
THAS745 - makes initial SEGWAY file.
THAS750 - makes initial Counter Locations file.
THAS752 - makes a Traffic Volume file for downloading to a PC.
THAS754 - creates the STATION.LIST file.
THAS755 - creates Station Volumes file.
THAS760 - creates Segment Volumes file.
THAS762 - implements a SubSegment-Counter Map file to create a SubSegment Traffic volume file.
THAS764 - implements the Node-Counter Map to create the Node traffic volume file.
THAS765 - makes initial SEGCLASS file.
THAS766 - combines two subsegment traffic volume files.
THAS768 - fills in missing nodes in the node traffic volume file.
THAS770 - adds HSNTAB sequence numbers to SEGCLASS file.
THAS775 - adds HSNTAB sequence numbers to SEGDIST file.

Subroutines

See Appendix A.

Notes on the following program descriptions:

1. The acronyms in the Input and Output sections are the PL/I file names, and thus the DD names to be used in JCL executing the program.

2. The program descriptions were copied into this manual from the design specifications documents. Although the descriptions were updated during development, the programs may do more than the descriptions specify.

3. Programs THAS700, THAS745, THAS750, and THAS765 were designed for one-time use. Detailed descriptions of them are not given here.

9.5.1 THAS105 - Convert Location Codes from older to 1995 LKI

Purpose: to convert location codes in accident records produced according to an earlier version of the Landmark-Kilometer-Inventory into corresponding location codes fitting a newer version of the LKI.

Input:
- INACC - Input accident file
- SHNFIL - Segment - Highway - Node file
- SINDEX - Segment Index files (concatenated)
- LMATCH - Landmark Match List files (concatenated)

Output:
- OUTACC - Output accident file (converted)
- BADDAT - Bad locations file - accident records whose location codes could not be converted (prefixed by 'SEG' and segment #)
- SYSPRINT - Messages and record counts

Notes:

- The Landmark Match List contains, for each segment, a list of matching KMMARKs in the old and new LKI's. For each segment, the Segment Index table (within the program) contains the following information pertaining to the segment as a whole.

- A pair of effective dates; accidents occurring between these dates will be converted. A segment may occur several times in the Segment Indexes, with different effective dates.

- The position (in the Landmark Match List for a given pair of effective dates) of the first matched KMMARK pair belonging to the segment. (This field and the next one are generated by THAS105, rather than being stored in the Segment Index files.)

- The position of the last matched KMMARK pair belonging to the segment.

- Four flags:

  - SEGMENT_ADDED - TRUE if the segment was added to the newer LKI (did not exist in the older LKI).
  - SEGMENT_REMOVED - TRUE if the segment was removed from the newer LKI.
  - SEGMENT_UNCHANGED - TRUE if all KMMARKs of the segment are identical in the newer and older LKI's.
  - SEGMENT_HAS_OBSOLETE_LOCNS - TRUE if there are portions of the segment that no longer occur or have been significantly changed in the newer LKI. (Usually means that a portion of road has been straightened or removed.)

- There may be any number of Segment Index and Landmark Match List files, concatenated in the JCL. Also, one file may contain several Segment Indexes or Landmark Match Lists. Segment Indexes may be complete or incremental. (This is signalled by the report-missing-segments flag in the file -- YES indicates a complete Segment Index, where missing segments must be reported as errors.)

- If an accident's segment number occurs in two or more Segment Indexes (and

the accident occurred within their date ranges), the accident will be put through the specified conversions in the order in which the Segment Indexes occur in the Segment Index files.

- If a segment number is missing from either a complete Segment Index or the Segment-Highway-Node table, then the location code cannot be converted. In this case, the accident record will be written to the bad locations file BADDAT. The accident record will also be written to the BADDAT file if the converted location code would be at a position past the end of the segment, as defined in the newer LKI.

- The program THAS110 also saves accident records it cannot handle in the same file BADDAT.

Description:

- Read in the Segment-Highway-Node table.

- Read in the Segment Indexes and the Landmark Match Lists.
  Determine the first and last Landmark Match List records for each segment, and add them to the Segment Index.

- Continue until the input accident file is finished:
  Read an accident record.

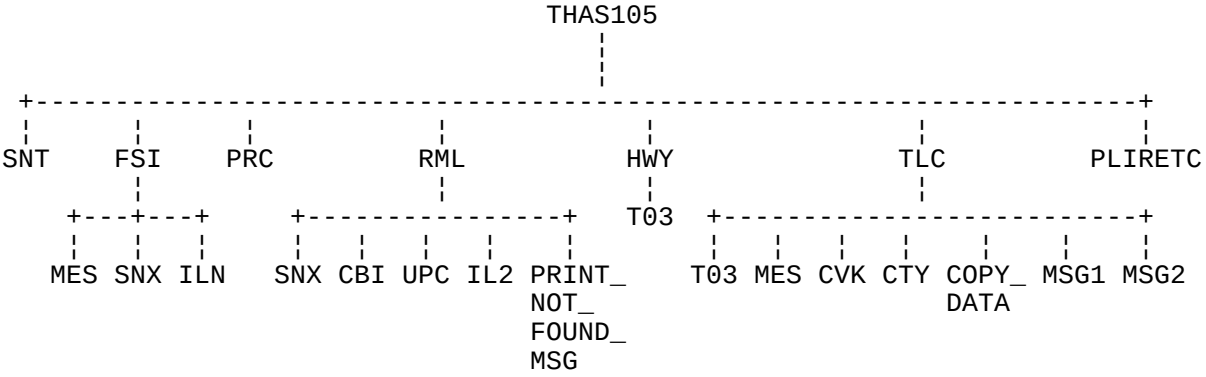
  For each Segment Index:
  If the accident's segment number occurs in this Segment Index, then:
  Translate its location code. (See section below.)

  Get the correct highway for the converted segment number from the Highway-Segment-Node table.

  If the accident's location code was converted successfully (it may become an obsolete location; this is considered a successful conversion) then:
  Write the accident record to the output accident file.
  Else:
  Write the accident record to the bad locations file, prefixed by the segment number.

- Count and report the number of records read from the input accident file and written to the output accident file and the bad locations file.

THAS105 Structure Chart



Utility: ABT

PRINT_NOT_FOUND_MSG, COPY_DATA, MSG1 and MSG2 are internal subroutines.

Details of Location Code Translation

```
If the accident date was not between the Effective Dates of the current
Segment Index
      OR the segment was added to the newer LKI
      OR the segment was unchanged
then:
      No translation is needed.

If the segment was removed from the newer LKI then:
      Mark the accident as happening at an Obsolete Location.
      No further translation is needed.

Check whether the accident's location corresponds exactly to the KMMARK of a
landmark in the older LKI.  If so, then:
      If the corresponding KMMARK in the newer LKI is blank, then:
            Mark the accident as happening at an Obsolete Location.
      Else:
      Copy the corresponding segment and KMMARK from the newer LKI into the
      accident record.

      If this matched pair of landmarks is marked as obsolete in the Landmark
      Match List, then:
                  Mark the accident as happening at an Obsolete Location.
      Endif

Else:  (the accident did not occur exactly at a landmark in the older LKI)
      Find the smallest i such that old_KMMARK(i) > the accident's KMMARK.

      If such an i is found, then:
            If all of:    { old_KMMARK(i), old_KMMARK(i-1),
                            new_KMMARK(i), new_KMMARK(i-1) }
            are non-blank, then:
                  Convert the KMMARK using the formula:
                        accident_KMMARK := new_KMMARK(i-1)
                              + ( accident_KMMARK - old_KMMARK(i-1) )
                                    * ( new_KMMARK(i) - new_KMMARK(i-1) )
                                    / ( old_KMMARK(i) - old_KMMARK(i-1) )
            Endif

            If new_KMMARK(i-1) or new_KMMARK(i) is blank, OR
              if pair i-1 or pair i is marked obsolete in the Lmk Match List,
            then:
                  Mark the accident as happening at an Obsolete Location.
            Endif

      Else:  (the accident KMMARK is > all the old_KMMARKs for its segment)
            If the accident's KMMARK can be converted to be < the new segment
            length using the formula:
                  accident_KMMARK := accident_KMMARK
                     - last matched old_KMMARK + corresponding new_KMMARK
            then do so.

            Else:
                  The location code cannot be converted; it will be written
                  to the Bad Locations file.
            Endif
      Endif
Endif
```

9.5.2 THAS106 - Convert Location Codes to the Current LKI

```
This program is an improved version of THAS105

Notes on conversion from THAS105 to THAS106:
 106:
    - requires match list to be pre-rounded to 1 decimal place
    - modified subroutines are: CVK renamed to CV6
                                RML renamed to RM6
                                TLC renamed to TL6
    - does interpolation with binary numbers instead of
      picture variables, for efficiency.
    - requires a cleaner match list: eg if there are > 1 matches
      at the same old kmmark, it is assumed that the 1st one can be
      used in all cases.
    - various other modifications made for efficiency.
```

9.5.3 THAS110 - Separate Segment and Node Data

```
Purpose:    to separate a sequential file of accident data into a file of
segment data, and a file of node data.

Input:      - INACC     - Input accident file
            - SHNFIL    - Segment - Highway - Node file


Output:     - SEGDAT    - Accidents in segments (with 8 byte prefixes)
            - NODDAT    - Accidents at nodes (i.e. at segment ends).
                          Each record is preceded by the node name.
            - BADDAT    - Records with segment-kmmark codes invalid,
according to the current LKI Segment file.
            - DATES     - Min and Max accident dates read.
            - SYSPRINT  - messages and record counts

Description:

-     Read the SHNFIL to create a table containing the following elements:
      SEGNUM, SEG_DATE, SEG_LEN, BEGIN_NODE, BEGIN_CONT, END_NODE, END_CONT

-     For each record on the input file:

       - Look for the segment on the table.
       - If Segment not on table, print message and write to BADDAT.

       - If Segment is on the table:

         Set LOCN_ID, Region, District and Day_of_week in the acc record.

         SELECT CASE:
           -      KMMARK=0
                  Translate the first character of Begin-node to alphabetic:
                    0123456789 to ZABCDEFGHI.
                  Write Begin_node and the accident record to file NODDAT.
           -      KMMARK=SEG_LEN
                  Translate the first character of End-node to alphabetic:
                    0123456789 to ZABCDEFGHI.
                  Write End_Node and the record to file NODDAT.
           -      KMMARK > SEG_LEN
                  Write the record to file BADDAT, and print a message.
           -      Otherwise:
                  Prefix the record with 'SEGnnnn ', where nnnn is the
                  segment number of the record.
                        Write to file SEGDAT.
```

9.5.4 THAS120 - Prepare for IEBGENER creation of Node PDS

```
Purpose:    to prepare a sequential file of accident data for input to
            IEBGENER, and to create the IEBGENER control statements for
            converting the sequential file into a PDS.

Input:      - IN        - file of accident data, sorted on the 8 character
                          prefix.
Output:     - OUT       - same as IN, but with the prefix blanked out on all
                          but the last record of each group.
            - GEN       - IEBGENER control statements.
Description:
   -  Read IN records, and copy to OUT, blanking out bytes 1-8 on all but the
      last record of each group of records with the same prefix.

   -  save prefix (member) names in a list.

   -  at the end, create GEN - the file of IEBGENER statements - from the
      list of member names.  Depending upon the JCL, this will be file
      SEGGENER or NODGENER, as follows:

File SEGGENER

First Record:

       GENERATE MAXNAME=<nsegs>,MAXGPS=<nsegs>,MAXFLDS=<nsegs>

where <nsegs> is the number of segments.

The following pair of records will be repeated for each segment:

       MEMBER NAME=SEGnnnn
      SEGnnnn  RECORD IDENT=(8,'SEGnnnn ',1),FIELD=(256,9,,1)

where nnnn is the segment number.

File NODGENER

First Record:

       GENERATE MAXNAME=<nnods>,MAXGPS=<nnods>,MAXFLDS=<nnods>

where <nnods> is the number of nodes.

The following pair of records will be repeated for each node:

       MEMBER NAME=Xnnnmmmm
      Xnnnmmmm RECORD IDENT=(8,'Nnnnmmmm',1),FIELD=(256,9,,1)

where Nnnnmmmm is the node name, and Xnnnmmmm is the node name with the first
digit translated to a character: Z for zero, and A-J for 1-9.

NOTE that the GENERATE and MEMBER statements must start with at least one blank.
```

9.5.5 THAS132 - Create Segment-Highway-Node File

```
Purpose: to create the segment-highway-node file from the
         segment file.


Input:      - SEGFILE   - Segment file

Output:     - SHNFIL    - Segment-Highway-Node file.
            - SYSPRINT  - messages and record counts


History: Prior to September 1999 the information now in the
         SEGMENT file was in the SEGMENT, SEGWAY, SEGNODE,
         and HWYSEG files.  These files are now combined into
         the SEGMENT file.

         SHNFIL contains the same fields as SEGMENT except for
         HWY2, HWY3 and SEG_REPORT_SEQ, which are not used
         in the Data Retrieval sub-system.

         This program has thus been reduced to simply copying
         the SEGMENT records into the SHNFIL records.
```

9.5.6 THAS134 - Create NODESEG File

```
Purpose:    Invert the Segment-Node file.

Input:      SEGNODE     - Segment-Node file

Output:     NODESEG     - Node-Segment file

Description:

   -  Read file SEGNODE into memory: into a SEGNODE table
   -  convert it into a NODESEG table containing two records for each SEGNODE
      record:     NODENAME, SEGMENT, B/E, C/D

      e.g.  2222 11112222 C 22223333 D     ... would become:

            11112222 2222BC
            22223333 2222ED


   -  sort the NODESEG table by node name
   -  scan the sorted NODESEG table, combining records with the same node
      name into a single NODESEG file record, then write the record out to
      NODESEG.
```

9.5.7 THAS720 - Modify Specified Accident Records

```
Purpose:    to copy an accident file inserting data into specified fields of
            specified accident records.

Input:      - INACC     - input accident file
            - FLDNAM    - THASP.TABLE(FLDNAMES)
            - SYSIN     - Each record of control input consists of an
                          accident case number, a page number ('*' for all
                          pages) a field name, and the replacement data for
                          that field.  All must be in quotes.

Output      - OUT       - Output accident file.
            - SYSPRINT  - echo of SYSIN,
                        - list of changes actually made

Description:

      - read in the control information

      - look up the position of each field in the record.

      - write out the control information

      - copy data, making changes.
```

9.5.8 THAS754 - Create STATION.LIST File

This program reads the counter station ID's from all the Counter-Map files, and from the COUNTER.LOCNS file, and writes a file containing a unique, sorted list of station IDs.

9.5.9 THAS755 - Create Station Volumes File

Purpose: to extract traffic volume data for all the count stations required by the H.A.S.

Input:
- INVOL - Input traffic volume file (sorted output from THAS753)
- STNLIST - Station List: list of all needed count stations. (output from program THAS754)

Output:
- STNVOL - Station Volumes file

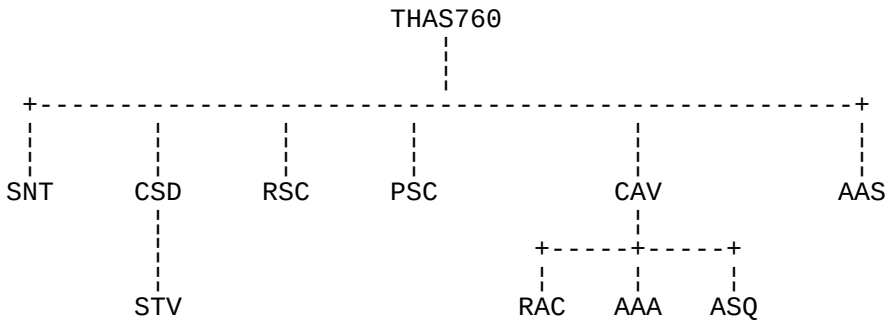
Description:

- Coordinate the INVOL and STNLIST files. For each counter on STNLIST, copy the traffic volume from INVOL to STNVOL.

9.5.10 THAS760 - Create Segment Volumes File

```
Purpose:    to average all available traffic volume data to create a single
            set of traffic volumes for each segment and year.

Input:      - DEFAULT   - traffic volumes defaults file
            - SHNFIL    - THASP.SHNFIL                Segment-Highway-Node
            - CNTLOCN   - THASP.COUNTER.LOCNS         Counter Locations file
            - STNVOL    - THASP.STATION.VOLUMES       Station Volumes file
            - SYSIN     - from-year and to-year as two 4-character strings

Output:     - SEGVOL    - Segment Volumes file

Description:

 -    Read input files.  Interpolate and extend year range of data using
      routine THASSTV.

For each year:
 -    If a segment has just one counter, use that data for the segment.

 -    If a segment has more than one counter, average the data, and use the
      average for the segment.

 -    If the segment has no counters, the previous and/or next segment
      qualifies for use if each one:
             - has one or more counts
             - is connected to and continuous with the current segment
             - is on the same Highway as the current segment

 -    If the current segment is one-way and a qualifying segment is two-way
      (or vice versa) halve (double) the counts of the qualifying segment
      before using.

 -    If both the previous and next segments qualify, use the average of
      their closest counts to the current segment.

 -    If only one of the previous or next segments qualifies, and the counter
      in the qualifying segment is within <MAX_COUNTER_DISTANCE> of the
      current segment, then use the data of that count for the current
      segment.

 -    If there are no qualifying adjacent segments, output no traffic volume
      records for the current segment.

 -    For each segment, print the stations on (or used for) the segment,
      their types, locations, and distance (if on adjacent segments), the
      annual ADT's for each year, and the number of missing monthly ADT's.
      This is the Segment Traffic Volume Report.

 -    Add a flag to the Segment Traffic Volume Report if there is a large
      difference (more than MAX_PERCENT_VARIANCE) between the annual ADTs for
      the segment.

 -    Print counts of segments with counters, segments with volumes
      calculated from adjacent segments with counters, segments with no
      available counters, and total number of segments.

Note: <MAX_COUNTER_DISTANCE> and <MAX_PERCENT_VARIANCE> are user-adjustable
parameters, defined in file THASP.DEFAULTS(TRAFFVOL).
```

THAS760 Structure Chart



Utility: TLKIDTR

9.5.11 THAS762 - Implement Subsegment-Counter Map

```
Purpose:    read a Subsegment or Intersection - Counter map file, extract the
            required data volumes from the STATION.VOLUMES file, and produce
            the traffic volumes for the specified sub-segments.

Input:      STNVOL      - Station Volumes file
            MAPFIL      - subsegment map file
            SYSIN       - from-year and to-year as four-character strings.

Output:     VOLOUT      - Sub-Segment Volume file

Description:

   -  For each subsegment record in MAPFIL:

         -   extract the volume data for the required years and count stations
             from STNVOL.  (Interpolate and extend year range of data as
             required using routine THASSTV).

         -   combine the volumes from the specified stations using the factors
             specified in the MAPFIL record.

         -   create and write a VOLOUT record for each year.

         -  Allow for negative weights in the Map file.
```

9.5.12 THAS764 - Implement Node-Counter Map

```
Purpose:    read a Node Counter map file, extract the required data volumes
            from the STATION.VOLUMES file, and produce the traffic volumes
            for the specified nodes.

Input:      STNVOL      - Station Volumes file
            MAPFIL      - node map file
            SYSIN       - from-year and to-year as four-character strings.

Output:     VOLOUT      - Node Volume file

Description:

   -  For each node record in MAPFIL:

         -   extract the volume data for the required years and count stations
             from STNVOL.

         -   combine the volumes from the specified stations using the factors
             specified in the MAPFIL record.

         -   create and write a VOLOUT record for each year.

   -  Allow for negative weights in the Map file.
```

9.5.13 THAS766 - Combine Subsegment Volume Files

```
Purpose:    Reads two sub-segment volume files, and overlays the data of one
            upon the other.

Input:      COARSE -    a sub-segment volume file (e.g. SEGVOL1)
            FINE   -    a more detailed sub-segment volume file (e.g.
                        SUBSEG.VOLUMES)
            SEGFIL -    Segment File (read so that a list of segments and
                        segment lengths is available)

Output:     VOLOUT -    FINE overlaid on COARSE

Description:

    Since the FINE data has precedence, the following algorithm takes the
    approach of copying the FINE data, and filling in any gaps with the
    COARSE data.

    When coordinating the files, the key used is Segment+Year+Start_km.

 THAS766 Algorithm:

    DECLARE: prev_fine and fine: records from the FINE file
             coarse:             record from the COARSE file

    prev_fine.end = 0

    Read Coarse

    Until Eof(FINE & COARSE)

        If Fine.start > prev_fine.end+0.1 then    /* gap in Fine */
            /* fill in from Coarse */
            While Coarse.End <= prev_fine.end     /* look for Coarse data to
                 read coarse                            fill the gap */
            end

            gapstart = prev_fine.end + 0.1
            gapend   = fine.start - 0.1

            /* fill the gap with coarse data */
            While Coarse.start < fine.start AND NOT eof(COARSE)
                START = max(gapstart,coarse.start)
                END   = min(gapend,  coarse.end)
                write Coarse data, with START and END
                read Coarse.
            end
        end

        if not eof(fine)
            prev_fine = fine
            Write fine (except first time)
            Read fine.  On endfile, set fine.start to a high value
        end
    end
```

9.5.14 THAS768 - Fill in Volumes for Un-mapped Nodes

```
Purpose:    Copy the NODE.VOLUMES file, calculating the volume for missing
            nodes using data from file SEGVOL2.

Input:      NODVOLM         - the node volumes created according to the
                              Node-Counter Map file
            NODLIST     -     a list of all node names
                              (e.g. file THASP.NODESEG)
            SEGVOL      -     a sub-segment volume file (e.g. SEGVOL2)

Output:     VOLOUT      -     volumes for every node

Description:

   -  coordinate NODVOLM and NODLIST, copying volume data to VOLOUT
   -  for each node on NODLIST which does not have data on NODVOLM,
      determine the volume for that node from the SEGVOL file:

         -  look up the volumes of the segments in the node name, using a
            kmmark 0.1 from the node, using routine THASVOL.
         -  average those two volumes to get the volume for the node.
```

9.5.15 THAS770 - Add HSNTAB Sequence Numbers to SEGCLASS File

Purpose: to add a sequence number to each record of the SEGCLASS file, preparing it to be sorted in search path order (by sequence number and KMMARK).

Input:
- SEGIN - SEGCLASS file
- SHNFIL - Segment-Highway-Node file (read into the HSNTAB)

Output:
- SEGOUT - SEGCLASS file with sequence numbers

Description:

- Read in the HSN table.
- Do until end-of-file:
  - Read a SEGCLASS record.
  - Find the index of its segment in the HSN table.
  - Add this index to the record and write the record to the output file.
- Print a count of records written to the output file.

9.5.16 THAS775 - Add HSNTAB Sequence Numbers to SEGDIST File

Purpose: to add a sequence number to each record of the SEGDIST file, preparing it to be sorted in search path order (by sequence number and KMMARK).

Input:
- SEGIN - SEGDIST file
- SHNFIL - Segment-Highway-Node file (read into the HSNTAB)

Output:
- SEGOUT - SEGDIST file with sequence numbers

Notes:

- Sections in the original LKI SEGDIST file are specified by segment, start-km, end-km, and offset. The offsets must be added to the start-km and the end-km to get true KMMARKs.

  Most of the offsets are zero, but some segments have sections that are specified with two different offsets. Also, some offsets are positive and some are negative, and both the offsets and the km's are in several different formats. The sorting on these segments would not be reliable.

  To save the situation, THAS775 adds the offsets to the start-km and end-km, and saves the new SEGDIST records with offset=0 and a consistent format.

- Input records are read with GET SKIP EDIT instead of READ statements. This allows THAS775 to be run on either the original LKI SEGDIST file (29-char records) or a sorted version of the file (33-char records). Only the first 29 characters of each record are read.

- At present, there is no utility job that runs THAS775. Eventually, it is intended that there be a single job that allows the user to edit any of several files and automatically does any necessary processing to prepare the file for use. This job would run THAS770, THAS775, or any other necessary programs.

Description:

- Read in the HSN table.
- Do until end-of-file:
  - Read a SEGCLASS record.
  - Find the index of its segment in the HSN table.
  - Add this index to the record.
  - If the start-km is blank, set it to zero.
  - If the end-km is blank, set it to 999.9.
  - If the offset is blank, set it to zero.
  - Add the offset to the start-km and end-km in a consistent format.
  - Set the offset to zero.
  - Write the record to the output file.

10 HASutil Visual Basic Application

See the User's manual for a description of the HASutil application.

HASutil is developed in Visual Basic 6.0.

At Cypher Consulting, the source files are located in folder:
\\CYPHER_XP\C_XP\Cypher\HAS\HASutil.

A copy of the source files (zipped) are kept on the Highways LAN in folder:
\\P:\HQ\ENG\safety\has\UTIL\NT\source, named **husrcnnn.zip** where 'nnn' is a version number.

The setup files, created using the VB6 Package and Deployment Wizard, are located in:
\\CYPHER_XP\C_XP\Cypher\HAS\HASutil\setup, and in:
P:\HQ\ENG\safety\has\UTIL\NT\setup

When running the Wizard:

- create a single CAB file,
- use the HASutil_Setup_Script (on CYPHER_XP),
- use the Standard Setup Package.

To update from one version of HASutil to the next it is often unnecessary to run the setup program. Simply copying the new files into the HASutil folder on the workstation eliminates the risk of the setup program replacing Windows system files! Update files are kept in the HASutil\UPDATE folder on the LAN.

11 HASLKI Maintenance

11.1 Introduction

The LKI files in \\CYPHER_XP\cypher\HAS\lki\HASLKI which correspond to the LKI files currently in production on the mainframe are:

```
HASLKI_DATA.MDB    - MS-Access Database of data tables
HASLKI_CODE.MDB    - MS-Access file containing queries,
                     reports and VB code.
                   - use macro LinkTables to (re)link to the
                     tables in a DATA.mdb database.
HASLKI.XLS         - Excel version of HASLKI_DATA.
```

Previous versions of the LKI may exist in this folder, with date suffixes, e.g. HASLKI2000.MDB

In the future, old versions will be named: HASLKI_DATA_yyyymmdd, where yyyymmdd is the date that that version was REPLACED.

LKI updates and corrections should be compiled in a database called HASLKInew_DATA.mdb:
only 1 HASLKInew_DATA.mdb file should exist
may be located wherever LKI work is being done
currently (Oct 22, 2002) it is in C_NT:\has\newlki

Utility code, queries, reports, and other code under develepment should reside in a database called HASLKI_UTIL.mdb, in the same folder as HASLKInew_DATA.mdb.

11.2 Implementing a new LKI

Create Match Lists to define location code transformations (if necessary)
Update/add traffic volume counter maps for changed/new segments

11.2.1 Update HASLKI_DATA.mdb

Rename \\CYPHER_XP\cypher\has\lki\haslki\HASLKI_DATA.mdb to
HASLKI_DATA_yyyymmdd.mdb
Copy over HASLKInew_DATA.mdb to \\CYPHER_XP\cypher\has\lki\haslki and rename to
HASLKI_DATA.mdb
Clean out HASLKI_DATA.mdb of extra tables used in updating information.
Check for new module code, reports and queries in newlki\HASLKI_UTIL.mdb which
should be copied into HASLKI_CODE.mdb (e.g. new or revised export functions.)
Check that Export functions to create text files of the Tables run correctly.
Use ExportAll function to create text files of the Tables :
Set CONST Outdir to \\CYPHER_XP\cypher\has\lki\haslki
run: Set prefix to yyyymm_ (effective date)

11.2.2 Upload exported text files to THASD files on mainfame

Note: this step should be done before uploading PDF and other files to the LAN, because testing on the mainframe can reveal errors in the LKI files.

- see uplki.bat
- check that those THASD files on the mainframe are not migrated (go to option 3.4, and if migrated, issue TSO HRECALL command)
- run uplki.bat
- upload revised counter map files to the mainframe (see volume section)

11.2.3 Test in THASD on the Mainframe

- add new match files to THASPDSM.SKL, and upload to THASD.
- run job PDSM in development, using production master input files.
- check PDSM output for error messages.
- run job PDSI when PDSM checks out OK.
- in HAS / Utilities menu, run jobs to sort SEGCLASS.YR1987, SEGCLASS.YR2002, SEGAREA (must be done after PDSM because they use SHNFIL created by PDSM).
- run volume generation in THASD - jobs 760 (maybe) and 761 - and check output.
- implement new volume files (761 creates *.new files)
- Run a data retrieval job in THASD to test. Begin and end node information can be tested by selecting data through a revised area, and checking the search path for discontinuities.

11.2.4 Create pdf copy of HASLKI_DATA.mdb

Look in C_NT:\has\LKIpdf\ folder for the pdf files needed to make complete report.
In HASLKI_CODE.mdb:
    Open Report : LKI_Book and make sure source of report is Query : Rep_All or appropriate query for area desired.
    Run Report : LKI_Book
Print to pdf writer
Check Table : SegPage lists the correct segments before running the following index reports.
Run IndexByReportSequence report.
Print to pdf writer
Run IndexBySegment report.
Print to pdf writer
Update TitlePage.doc
    Make sure to check page 2 Landmark Types is up to date.
    If necessary, cut and paste updated Landmark Types from the table in HASLKI_DATA
Print to pdf writer.
Combine files in order:
    Title Page
    IndexByReportSequence
    IndexBySegment
    LKI_Book
To combine 2 PDF files:
    With target document open, choose Document > Insert Pages.
    In Select File to Insert dialog box Select source document to insert into Target document,
    Specify whether you want to insert Before or After a specified page.
Repaginate so indexes point to correct page.
    Click the Show/Hide Navigation Pane Button on Menu Bar
    Click Thumbails to bring palette to the front
    Right-click on pane to bring up pop-up box and choose Number Pages
    Specify page range (all pages up to beginning of LKI_Book)
    Choose Style i, ii, iii
    Make sure page number 1 is first page of LKI_Book.

11.2.5 Create HASLKI.xls file

- Create Excel file using AllToExcel macro in HASLKI_CODE.mdb
- Creates a file named haslki.xls in the current folder.

11.2.6 Upload HASLKI files to the LAN

In folder P:\HQ\Eng\safety\has\lki,

- copy existing HASLKI files down to the OldVersions folder, renaming using their expiry date

- upload new HASLKI.XLS, LKI_BC.PDF, HASLKI_DATA.MDB and HASLKI_CODE.MDB
- notify Gord Smith at Gord.Smith@gems8.gov.bc.ca, that the web copy of LKI_BC.PDF http://www.th.gov.bc.ca/publications/eng_publications/geomet/lki/LKI_BC.PDF should be replaced from the copy on the LAN.

11.2.7 Go into production on the Mainframe

- rename current THASP LKI files & SHNFIL - use THASD.REXX.EXEC(RNAMLKI)
- promote all LKI files - use THASD.REXX.EXEC(PROMLKI)
- promote new match and segment list files.
- promote all counter map files - use THASD.REXX.EXEC(PROMAPS)
- promote new PDSM skl (with new match list files added)
- promote new version of THAS106 (if it was modified)
- run job PDSM, using production master input files.
- check PDSM output for error messages.
- run job PDSI when PDSM checks out OK.
- in HAS / Utilities menu, run jobs to sort SEGCLASS.YR1987, SEGCLASS.YR2002, SEGDIST (must be done after PDSM because they use SHNFIL created by PDSM).
- run volume generation - jobs 760 (maybe) and 761 - and check output.
- implement new volume files (761 creates *.new files)
- Run a data retrieval job to test. Begin and end node information can be tested by selecting data through a revised area, and checking the search path for discontinuities.

12 Converting Location Codes for LKI Changes

12.1 Summary

12.1.1 Introduction

Whenever either a new version of the Landmark-Kilometre Inventory (LKI) is released (a complete conversion) or minor changes are made to the highways requiring modifications to the current LKI (an incremental conversion), there will be location codes in the earlier LKI which either refer to a different location in the new LKI or else is no longer a valid location.

Locations codes are kept as originally coded on the Tape Master file. Location codes are transformed to the latest LKI when the PDS-Master files are created, in utility Job PDSM. Where an accident occurred on a section of road that no longer exists as part of the highway system, the accident is either thrown out, or the location code is transformed (as sensibly as possible) and then marked as an obsolete location.

12.1.2 The Transformation Technique

For each transformation, a 'Match' file is created, in which landmarks of the older LKI are matched with landmarks on the newer LKI. Each match file is prefaced with a date range indicating the accident records to which the transformation should be applied. The transformation programs (THAS105 & 106) search for an accident's location in the match file. If found, the new location is substituted. If it is not found, the new location is obtained by interpolating from the spanning match records.

An LKI Landmark File contains over 25,000 landmarks, so creating the match file for a complete conversion is not a trivial matter. No attempt is made to match the landmarks automatically using the landmark descriptions. A two-column 'OLDNEW' file is created with the landmarks of the old and new landmark files in the left and right columns. A PC-based text editor with column capablities (PCWrite) is used to manually line up horizontally locations and descriptions of the same landmarks. The Match file is created from the resulting matched OLDNEW file.

12.1.3 History

The LKI transformation system was devised and implemented for the conversion to the January 1990 LKI. This was done by by Matthew Nicoll and Mia Shinbrot of Cypher Consulting, and Elke Pagel of the Highway Safety Branch.

Work on converting to the the April 1995 LKI was begun by Elke Pagel and Paul Gerber (ISB). It was finished by Matthew Nicoll, after Elke moved on to MVB, and Paul moved on to Finance. PC files relating to this conversion were archived on the LAN, on the Safety P drive, under directory:

P:\HAS\LKICNV95

12.1.4 File Naming

The files containing old and new landmark location codes and descriptions in left and right columns have a number of names:

```
OLDNEW files
MATCH files
TRANSFORMATION files
SEGMENT files (in program MAKELMK)
```

The file produced by the MAKELMK program which contains just the old and new landmark location codes (no descriptions) is consistently called the Match List file.

12.2 Steps for a Complete Conversion

This is NOT a simple automatic procedure! It requires a programmer with knowledge of the H.A.S. system on the mainframe, knowledge of PC programming, and a good understanding of the LKI conversion process. The landmark matching can be done by anyone with a good understanding of the LKI, and some PC experience.

(Source and executable files are on the LAN in P:\HQ\ENG\safety\HAS\LKI\util)

1. Make sure all the User-Maintained Highway Definition files (listed in Appendix A of the User's Manual) are converted to the new LKI. It may be possible to generate some of these files from whatever database the new LKI is defined. Currently (2001), the HAS LKI is maintained in an MS-Access database called HASLKI

   PL/I programs THAS161-164 may be useful for converting the SEGDIST, SEGCLASS COUNTER.LOCS and COUNTER.MAP files. NOTE: these programs only handle the simple cases properly - go through manually to fix up reversed segments, split and joined segments.

   Note: See macros and modules in the HASLKI database for exporting to text and csv files.

   These files can be worked on concurrently with following steps 2-10.

2. Create a two-column Old-New landmark match file, with landmarks of the old LKI in the left column and the new in the right column.

   There is no rigorous documentation of the match file format. See the Pascal program which reads them: MAKELMK.PAS.

   PL/I program THAS700 was used to create the OLDNEW file for the 1990 LKI conversion.

   For the 1995 conversion, the OLDNEW file was created using Export facilities from a MS-Access database containing both the old and new landmark files. The database is (was!) on the LAN: Safety P:\HAS\LKICNV95\ACCESS\ACCESS.ZIP

   Note that file ED.LMK, included with this PCWrite installation, causes PCWrite to be configured with the two columns of the OLDNEW files whenever a file with the .LMK extension is loaded.

   ```
   *****  SEGMENT 1234  *****
   ***  Old Landmarks                    ***  New Landmarks
   ```

3. If necessary, download the OLDNEW file from the mainframe to a PC.

4. Run Pascal program LKISPLIT to break the OLDNEW file down into smaller files so that they can be conveniently worked on on the PC. The files must be named with an extension of LMK, e.g. OLDNEW.LMK

For the 1995 LKI, 35 files of about 60 k each were created. Make sure they break at segment breaks. It might be worth modifying LKISPLIT to add a PCWrite 'column break' code (Ascii 11, Ascii 15) between segments.

This will streamline the matching process. Program MAKELMK is programmed to ignore PCWrite code lines.

5. Install PCWrite (if necessary)
   - create a PCWRITE directory on the hard drive of the PC,
   - from the PCWRITE directory, copy PCWrite in from the Lan:
     P:\HAS\LKICNV95\PCWRITE\PCWRITE.ZIP (old)
     or P:\HQ\ENG\SAFETY\HAS\LKI\UTIL\PCW3.ZIP (2004)
   - in AUTOEXEC.BAT, add the PCWRITE directory to the path, and define environment variable PCWRITE to point to the PCWRITE directory.

6. On the same PC, install Pascal programs MAKELMK and FIXLML and batch files ML1.BAT and MLALL.BAT

   These programs are in P:\HQ\ENG\SAFETY\HAS\LKI\UTIL\ on the LAN. (Before LAN reorg, they were in Safety P:\HAS\LKICNV95\MATCH\UTILS.ZIP). Modify (or re-write) the batch files to suit prevailing file names, preferences etc.

   The Pascal programs are compiled with a DOS version of Borland Pascal: Turbo Pascal 5.0.

7. Use PCWrite to line up corresponding landmarks in the OLDNEW match files. (See instructions and examples in following section: "Matching Landmarks in the Match Files")

   Run batch file ML1 after finishing each OLDNEW file.
   Check the log files which warn of possible errors in the matching.
   Repeat until satisfied that the OLDNEW file is correct.

8. When all the OLDNEW files are matched:

   Edit the MLALL.BAT file and correct the date range.
   The date range must identify the accidents to which you want the transformation to be applied.

   Run batch file MLALL.BAT to process ALL the OLDNEW files, creating one output landmark match list and its segment index file. (MATCH.LST & SEGIND.LST)

   SEG.LOG file contains information about each segment in following format:

```
Segment 0322
Segment 0324
Segment 0346    ADDED
Segment 0348                                    HAS OBSOLETE LOCATIONS
Segment 0351
Segment 0352                        UNCHANGED
Segment 0353                                    HAS OBSOLETE LOCATIONS
Segment 0354            REMOVED
Segment 0355
Segment 0356                        UNCHANGED
```

```
   Segment 0357                          UNCHANGED
   Segment 0359    ADDED
```

9. Upload MATCH.LST and SEGIND.LST into mainframe files:
   THASD.LANDMARK.MATCH.LIST.<id>
   THASD.SEGMENT.INDEX.<id>
   where <id> identifies the new LKI (eg. APR95).

10. Modify the JCL skeleton for job THASPDSM.
    - Add concatenated DD statements to the THAS106 step for the new LANDMARK.MATCH.LIST and SEGMENT.INDEX files.
    - Consider removing the THAS105 and THAS152 steps if accident prior to Jan 1990 are no longer included in the PDS-Master.

11. Run Utility job PDSM (in development) to recreate the PDS-Master files. Check the output of this job for indications of problems in the LKI or match files.

12.3 Incremental Conversions

12.3.1 Introduction

Incremental landmark conversions must be coded whenever changes are made to the LKI, between complete conversions.

To implement an Incremental Conversion, Incremental Landmark Match Lists and Segment Indexes must be coded, and concatinated to the main Landmark Match List and Segment Index in the THAS106 JCL step of job THASPDSM.

The incremental conversions done prior to the April 1995 complete conversion are coded in THASP.SEGMENT.INDEX.INCREM and THASP.LANDMARK.MATCH.LIST.INCREM. (These files are defined the **Files** section of this Manual.)

The steps for an incremental conversion are the same as for a complete conversion except the long process of producing the match list files (steps 2 to 10) are replaced by coding them directly - this being feasible for the small job of modifying one or two segments. As of October 2004, another option is to the matching in Excel (details below).

12.3.2 Effective-date ranges

The two dates given on an effective-date range line are the range of dates (inclusive) within which a particular set of Location Code transformations should be done. They are in the form YYYYMMDD

If a given set of road and/or LKI changes were complete by a certain date (say 31-December-1993) and it is desired to transform all accidents on or before that date, use an effective-date range like this:

```
19800101 19931231
```

(The first date, 1-January-1980, was chosen to be before any accidents remaining in the Master Files.)

To apply a transformation to all accidents on or after a certain date (say 3-April-1991), use a date range like this:

```
19910403 20301231
```

Transformations are performed in the order they occur in the Segment Index file(s). Thus, to perform a transformation on accidents before 1990, then another transformation on all accidents (say to correct an error in the 1990 LKI), then a third transformation on accidents before 1993, (i.e. through the end of 1992), the consecutive effective-date ranges in the Segment Indexes should be:

```
19800101 19891231
19900101 20301231
19800101 19921231
```

The third date range begins in 1980 rather than 1990 because pre-1990 accidents should be put through both the first and third transformations in turn.

12.3.3 Incremental Landmark Match Lists

See the definition of the Landmark Match List file in the **File** section.

This file may be coded directly, or an Excel utility may be used, as described below.

Following is an example match list. Be careful to use the correct columns.

Line up decimal points in columns 9 and 23.

There is no need to include every landmark on unchanged sections of a segment. In the example below, the sections of road between KMMARKs 0.00 and 4.96, and between KMMARKs 5.86 and 15.71, are unchanged. There may be many landmarks in these sections of road, but there is no need to include them in the Landmark Match List. Note also that there are no gaps before or after the obsolete section of road, in either the left or the right column.

```
19870101 19970101
1234   0.00   1234   0.00
1234   4.96   1234   4.96
1234   4.97 ! 1234   4.97
1234   5.85 ! 1234   9.02
1234   5.86   1234   9.03
1234  15.71   1234  18.88
```

Note: as of 2003-03-05, there is no need to round the kms in the landmark match list. PL/I procedure THASRM6 was originally written assuming that kms were rounded to 1 decimal place. Un-rounded kms in match lists entered to THAS106 will have caused some in-accurate transformations prior to March 2003.

Matching in Excel:

- Copy the old segment landmark data from Access into columns A-F
- Leave column G for the obsolete location marker.
- Copy the new segment landmark data from Access into columns H-M
- Open file MatchMacros.xls, in the same instance of Excel, (but a separate workbook.)
- To move data up or down, put the cursor on the top row to be moved, anywhere in the section to be moved (old, obs or new areas), and press Ctrl-D to move down, and Ctrl-U to move up.
- Run macro WriteMatchlist to export to a match list file.
- Edit the match list file with a text editor, if desired, to remove sections of redundant matches.

12.3.4 Incremental Segment Indexes

See the definition of Segment Index files in the **Files** section.

Make sure that each Segment Index begins with an effective-date range line with exactly the same dates as the corresponding Landmark Match List.

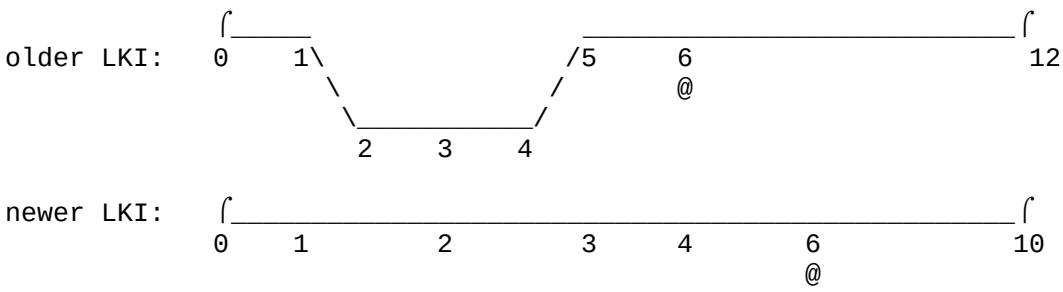
A report-missing-segments line may be included after the effective-date range line. If this line is present in an incremental Segment Index, the flag (at the beginning of the line) must be NO. This is the default, in any case, so this line is not necessary for incremental files.

The Segment Index must include a data line for every segment mentioned in either column of the corresponding Landmark Match List. There may also be data lines for other segments, where the segment-added or segment-removed flag is set; such segments need not appear in the Landmark Match List.

The four flags in the Segment Index are segment-added, segment-removed, segment-unchanged, and segment-has-obsolete-locations. Each flag may be zero (if the corresponding condition is false) or one (if the condition is true). There should be no need to include unchanged segments in incremental transformation files, so the segment-unchanged flag should always be zero. Of course, the segment-added and segment-removed flags should not both be set to one for the same segment. The segment-has-obsolete-locations flag should be set to one if the segment occurs in the left-hand column of the Landmark Match List with any exclamation points (for obsolete locations). This last flag is not used at present, but may be needed at some time in the future.

12.4 Segment Modification Scenarios

12.4.1 A portion of a segment is straightened:



Problems if no conversion were done:

In the above example, the road has not changed between kmmarks 5.0 and 12.0; only the measurements are different.

An accident that occurred at kmmark 6.0 of the older LKI (marked with an @ sign) would have occurred at kmmark 4.0 if measured from the newer LKI. If its location code were not converted, it would still appear at kmmark 6.0, which is an entirely different location.

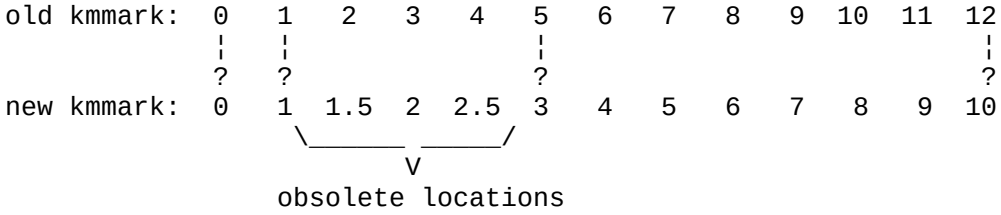
Also, accidents which occurred between kmmarks 10.1 and 12.0, left unconverted, would seem to have been coded incorrectly and would not be saved in the PDS Master Files, since their kmmarks would be greater than the new segment length.

Accidents which occurred between kmmarks 1.1 and 2.9, the first half of the removed section, would appear at kmmarks 1.1 - 2.9, which is the entire new straightened section.

Accidents which occurred between kmmarks 3.1 and 4.9, on road that no longer exists, would appear at kmmarks 3.1 - 4.9 of the unchanged section of the highway, which is not at all where they actually occurred.

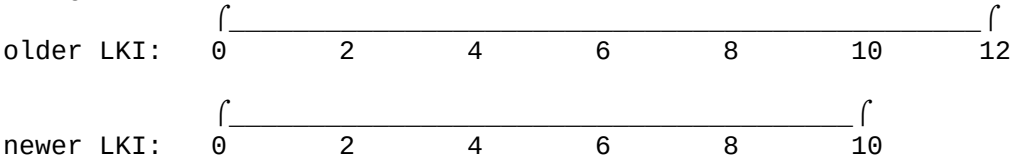
Solution chosen:

Accidents occurring between kmmarks 5.0 and 12.0 will be converted to kmmarks 3.0 through 10.0, where they belong. Accidents occurring in the straightened section, between kmmarks 1.1 and 4.9, will be compressed into kmmarks 1.1 through 2.9 and marked as obsolete locations. The mapping is shown below.

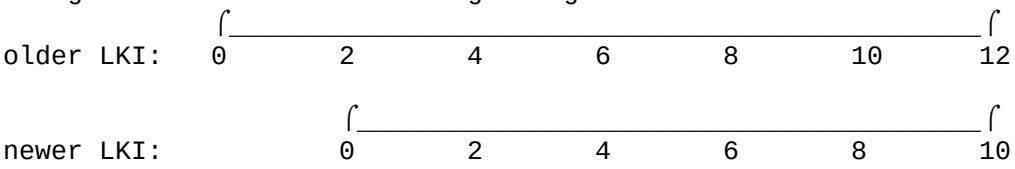


12.4.2 A segment is shortened at its beginning or end:

A segment shortened at its end:



A segment shortened at its beginning:



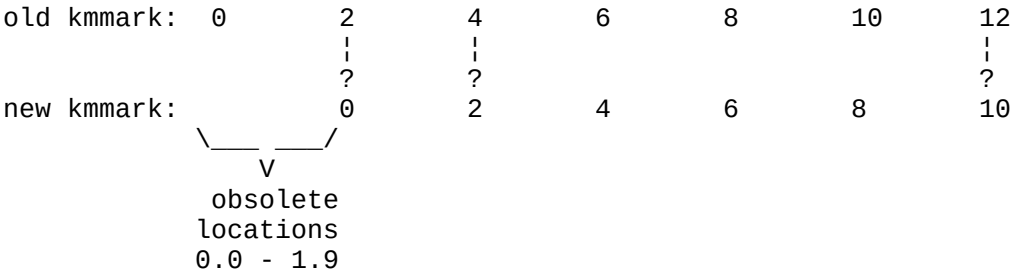
Problems if no conversion were done:

The segment shortened at its end is not a problem. The segment shortened at its beginning, however, is a major problem, since kmmark 2.0, for instance, is an entirely different location depending on whether an accident was reported using the old or the new LKI.

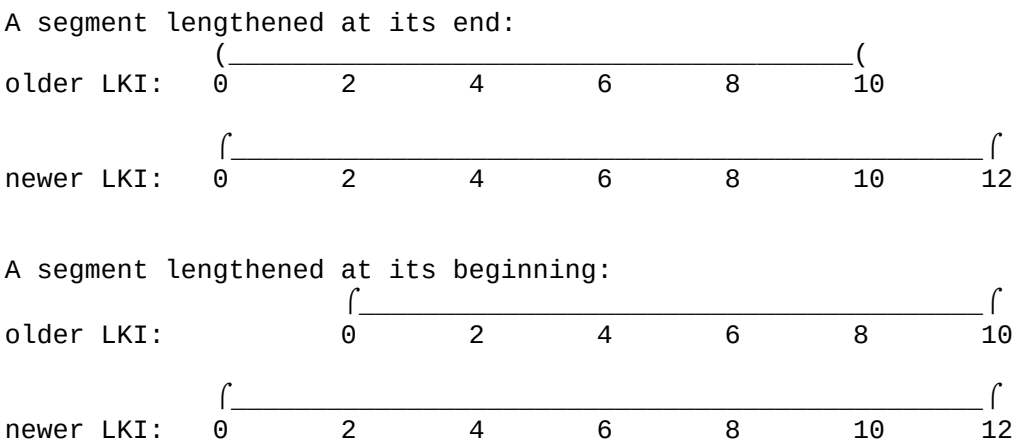
Solution chosen:

When a segment is shortened at its end, accidents occurring after the new end node (kmmarks 10.1 - 12.0 in the above pictured case) will be marked as obsolete locations and will not be saved in the PDS Master Files. They will still be on the original tape file, and in addition job PDSM will save them in the Bad Locations File THASP.THASPDSM.BADLOCN. Accidents occurring at the new end node (kmmark 10.0) will be considered as happening at a node, whether they occurred before or after the segment was shortened.

When a segment is shortened at its beginning, the accidents occurring on the removed section of road (kmmarks 0.0 - 1.9 above) will be marked as obsolete locations, and saved in the PDS Master Files. All other accidents occurring before the Effective Date of the new LKI will be mapped to the corresponding locations on the new LKI. The mapping is shown below.



12.4.3 A segment is lengthened at its beginning or end:



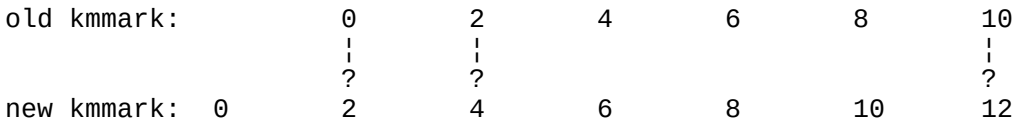
Problems if no conversion were done:

The segment lengthened at its end is not a problem. The segment lengthened at its beginning, however, is a major problem, since kmmark 2.0, for instance, is an entirely different location depending on whether an accident was reported using the old or the new LKI. Also, accidents occurring at the end node (old kmmark 10.0 above) would not be recognized as being at a node, since kmmark 10.0 is not a node in the new LKI.

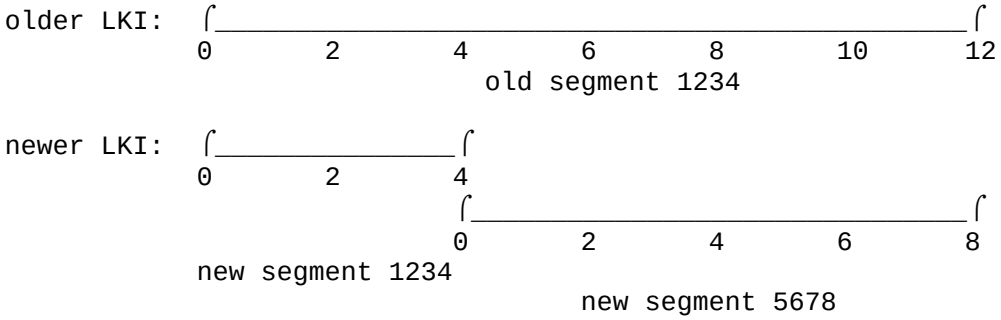
Solution chosen:

When a segment is lengthened at its end, we might not expect accidents to have occurred on the new section. However, the road changes were almost certainly made before the new LKI was released, and the RCMP may have started reporting accidents more-or-less accurately with kmmarks measured from where the old end node used to be. Therefore, accidents with kmmarks greater than the old segment length but within the new segment length will be accepted unchanged.

When a segment is lengthened at its beginning, accidents occurring before the new LKI's Effective Date will be mapped to the corresponding locations on the new LKI. The mapping is shown below.



12.4.4 A segment is split into two segments:



Problems if no conversion were done:

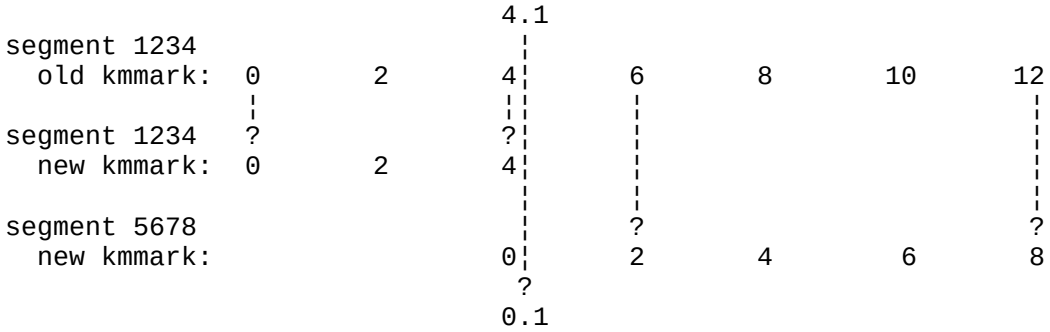
All accidents occurring after the new end node of the original segment (kmmark 4.0 above) would be considered to be incorrectly coded and would not be saved in the PDS Master Files, since their kmmarks would be greater than the segment length in the new LKI.

If the original segment name were retained for the last part of the segment instead of the first part, the situation would be even worse, since it would be similar to the case of a segment shortened at its beginning described above.

If the segment were given two new segment names, all accidents occurring on the segment before the new LKI's Effective Date would be removed from the PDS Master Files, since the old segment name would no longer appear in the LKI.

Solution chosen:

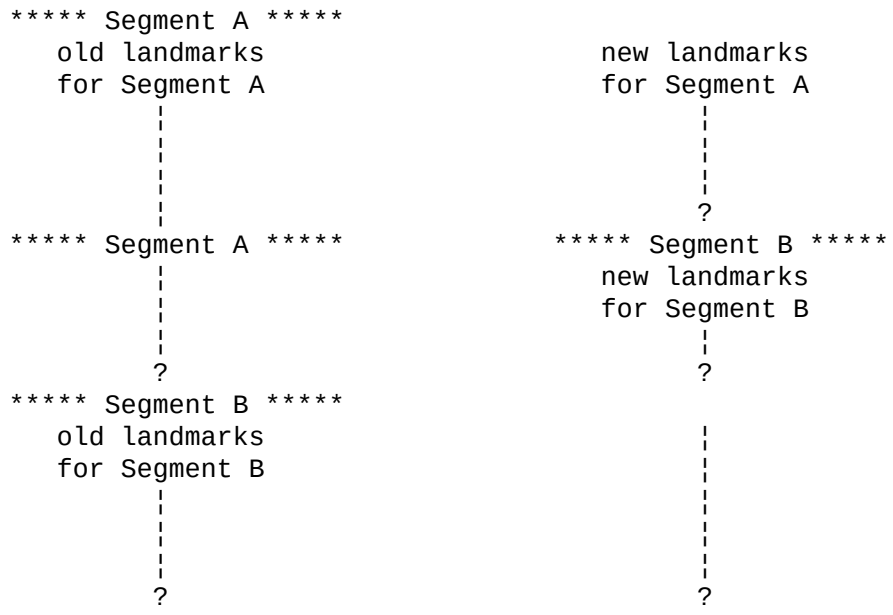
Accidents occurring before the new LKI's Effective Date will be mapped onto the two new segments. The segment names in the accident records will also be changed to fit the new LKI. The mapping is shown below.



That is, kmmarks 0.0 - 4.0 of the old segment 1234 are mapped into kmmarks 0.0 - 4.0 of the new segment 1234, and kmmarks 4.1 - 12.0 of the old segment 1234 are mapped into kmmarks 0.1 - 8.0 of the new segment 5678.

12.4.5 A Node Has Moved

If a node is moved, the combined Transformation file should be laid out like this:



12.5 Matching Landmarks in the Match Files

12.5.1 Object

Each Match file (produced by PL/I program THAS700) contains two columns of landmarks (KMMARK and landmark description). The first column holds the landmarks from the older version of the LKI. The second column holds the landmarks from the newer version of the LKI.

Use the DOS word processor PCWrite to edit the old-new landmark files. The object of the exercise is to line up **corresponding** old and new landmarks by inserting blank lines in one column or the other. Only the landmarks that are **matched** in this way will be used in the conversion.

Below is shown an example of an actual Match File from the 1980/1990 LKI's.

Before Matching:

```
+-------------------------------------------------------------------------------+
|   *****  SEGMENT 0314  *****                                                 |
|   *** Old landmarks:                    *** New landmarks:                    |
|     0.00 EAST SAANICH RD                 0.00 EAST SAANICH ROAD               |
|     1.38 HWY 17-PAT BAY HWY              0.53 SAANICH PENINSULA HOSPITAL      |
|     1.66 JCT COLLINS RD                  0.99 CENTRAL SAANICH ROAD            |
|     2.16 JAMES ISLAND RD                 1.37 HIGHWAY 17, PAT BAY HIGHWAY     |
|     2.29 ARTHUR RD                       1.64 MOUNT NEWTON CROSS ROAD ON LOCKS|
|     2.61 JAMES ISLAND FERRY              1.77 FERGUSON ROAD                   |
|                                          2.17 LOCKSIDE DRIVE ON JAMES ISLAND R|
|                                          2.30 ARTHUR DRIVE                    |
|                                          2.52 TURGORSE ROAD                   |
|                                          2.59 JAMES ISLAND FERRY WHARF        |
+-------------------------------------------------------------------------------+
```

After Matching:

```
+-------------------------------------------------------------------------------+
|   *****  SEGMENT 0314  *****                                                 |
|   *** Old landmarks:                    *** New landmarks:                    |
|     0.00 EAST SAANICH RD                 0.00 EAST SAANICH ROAD               |
|                                          0.53 SAANICH PENINSULA HOSPITAL      |
|                                          0.99 CENTRAL SAANICH ROAD            |
|     1.38 HWY 17-PAT BAY HWY              1.37 HIGHWAY 17, PAT BAY HIGHWAY     |
|                                          1.64 MOUNT NEWTON CROSS ROAD ON LOCKS|
|     1.66 JCT COLLINS RD                                                       |
|                                          1.77 FERGUSON ROAD                   |
|     2.16 JAMES ISLAND RD                 2.17 LOCKSIDE DRIVE ON JAMES ISLAND R|
|     2.29 ARTHUR RD                       2.30 ARTHUR DRIVE                    |
|                                          2.52 TURGORSE ROAD                   |
|     2.61 JAMES ISLAND FERRY              2.59 JAMES ISLAND FERRY WHARF        |
+-------------------------------------------------------------------------------+
```

12.5.2 Obsolete Location Markers

Section 7.1 gives some examples of the types of changes that would lead to a portion of road becoming obsolete.

Below is a part of the actual Transformation File for segment 0307 in the 1980/1990 LKI conversion. A portion of the highway has been moved and shortened. The entire changed portion has been marked as obsolete by placing an exclamation point (!) between the old and new landmarks. This exclamation point must be somewhere within columns 39-41 of the Transformation File.

```
+-------------------------------------------------------------------------------+
|    24.04 QUADRA ST.U/P    2435              23.67 QUADRA STREET, UNDERPASS     |
|    24.74 VANALMAN AVE                       23.87 VANALMAN AVENUE              |
|                                             24.57 LILY AVENUE                  |
|                                             24.90 PEDESTRIAN OVERPASS          |
|    25.43 ***elke                            25.21 ***elke                      |
|    25.44 ROGERS AVE       2458          !   25.22 PEDESTRIAN OVERPASS          |
|    26.25 MCKENZIE AVE                   !                                      |
|                                         !   26.05 SEVEN OAKS STREET, LEFT      |
|                                         !   26.14 CANTERBURY ROAD              |
|                                         !   26.24 RALPH STREET                 |
|                                         !   26.35 KENT ROAD                    |
|                                         !   26.44 HAYNES ROAD                  |
|                                         !   26.57 ACCESS TO SAANICH FIRE DEPT, RIG |
|    27.20 VERNON AVE                     !   26.79 VERNON AVENUE                |
|    27.41 CAREY RD                       !                                      |
|    27.46 RAVINE BRIDGE                  !   26.93 ***elke                      |
|    27.47 ***elke                            26.94 RAVINE WAY                   |
|                                             27.47 ROAD RIGHT (ACCESS TO SAVE-ON)   |
|                                             27.61 SAANICH ROAD, SOUTH          |
|    27.87 SAANICH ROAD                                                          |
|                                             27.70 SAANICH ROAD, NORTH          |
|    28.41 CLOVERDALE AVE                     27.73 CLOVERDALE AVENUE            |
+-------------------------------------------------------------------------------+
```

The landmark matches in this part of the file are:

```
            24.04 ?---? 23.67
            24.74 ?---? 23.87
            25.43 ?---? 25.21
            25.44 ?-!-? 25.22
            27.20 ?-!-? 26.79
            27.46 ?-!-? 26.93
            27.47 ?---? 26.94
            28.41 ?---? 27.73
```

Note that in some cases an extra landmark has been inserted in one or both columns (with descriptions of ***elke to indicate who put them in). This must be done to ensure that the first match of the obsolete section is **immediately** after the last match of the normal section, and the last match of the obsolete section is **immediately** before the first match of the next normal section.

12.5.3 Unchanged, Added and Removed Segments

For Unchanged, Added or Removed segments, delete all the landmark lines from the match file, and modify the ******* SEGMENT 1234 ******* header line by replacing the second set of asterisks with the keyword UNCHANGED, ADDED, REMOVED or OMITTED. The purpose of doing this is primarily to make the Utility Job PDSM run faster.

The line must start with at least 3 asterisks.

1) Removed segments.

******* SEGMENT 1234 REMOVED.**

This special-characteristic line is to be used when a whole segment has been removed from the new LKI. It is not to be used when a segment has been renumbered.

Accidents that occurred on removed segments will be excluded from the PDS-Master.

2) Added segments.

******* SEGMENT 1234 ADDED.**

This special-characteristic line is to be used when a whole segment has been added to the new LKI. This may be newly-constructed highway, or road that used to be part of municipal jurisdictions. It is not to be used when a segment has been renumbered or split in two.

Accidents that occurred on added segments will be accepted as is -- no Location Code conversion is possible or necessary.

3) Unchanged segments.

******* SEGMENT 1234 UNCHANGED.**

This special-characteristic line is to be used when a whole segment and all the landmarks in it are completely unchanged. It is not to be used when the segment has been re-measured and some of the kmmarks of its landmarks have been changed even slightly.

Accidents that occurred on unchanged segments will be accepted as is. This is much faster than it would be to go through the conversion process, only to come out with the same kmmark that went in.

4) Omitted segments.

Note October 1996: This feature (and documentation) was not verified in the 1995 conversion.

******* SEGMENT 1234 OMITTED FROM THE 1990 LKI.**

This special-characteristic line has been provided to deal with unchanged segments that were accidentally left out of the LKI. There are such segments in the 1990 LKI, and may happen to be some in later LKI's as well. The keyword **OMITTED** must be in the line; other information, such as **FROM THE 1990 LKI**, can be added for the sake of human beings who may examine the Location Code Transformation files.

Omitted segments will be treated exactly like unchanged segments, except that the PDS Master Files are deliberately constructed to conform to the current LKI. Therefore, accidents in omitted segments will not be saved in the PDS Master, but will be placed in the Bad Locations File instead. To ensure that such accidents go in the PDS Master File, you will have to add the segments to the new LKI (unchanged from the way they were in the older LKI) and rerun Utility Job PDSM. This can be safely done, without side-effects on the rest of the new LKI.

12.6 Using PCWrite

12.6.1 Starting PCWrite to Edit a Match File

The old shareware PCWrite editor was chosen for this task because it has column-editing capabilities, but produces straight ASCII output files. I.e. it does not insert any special formatting characters.

PCWrite must be installed on your PC, with its directory in your path, and also in the PCWRITE environment variable. File ED.LMK in the PCWrite directory contains the column definitions for the columns old-new landmark match files. The files must have the .LMK extension for the column definitions to be automatically loaded.

For simplicity, the old-new landmark match files are just called 'match' files for the remainder of this section.

To edit a match file (e.g. match01.lmk), get to the DOS prompt in the directory containing the match files, and type:

ed match01.lmk /s

The **/s** option tells PCWrite to make a backup copy of the file, named with an extension starting with **&**. In the above example, the backup file name would be match01.&lm.

To exit PCWrite and save the file you're editing:

Press function key <F1>, then <F2>.

To exit PCWrite WITHOUT saving your changes:

Press <F1>, then <F9>, then <F2>.

To save a file and continue editing it:

Press <F1>, then <F3>.

12.6.2 PCWrite commands

Here are some basic PCWrite editing commands:

To move around the file one line or one space at a time:
Use the arrow keys.

To move back and forth between columns:
Use <Shift>-<Right arrow> and <Shift>-<Left arrow>.

To insert a line within a column:
Move the cursor to the end of the previous line or the beginning of the next line and press <Enter>.

To delete a blank line in a column:
Move the cursor onto that line and press <Delete>.

To rejoin a line that you have accidentally split (by pressing <Enter> in the middle of the line):
Move the cursor to the end of the first half of the split line and press <Delete>.

There are a few other commands that are useful in moving around a file quickly.

To scroll the screen up or down without moving the cursor:
Use the <PgUp> and <PgDn> keys.

To move up or down a screen-full at a time:
Use <Shift>-<PgUp> and <Shift>-<PgDn>.

To move to the top or bottom of the current column:
Use <Shift>-<Up arrow> and <Shift>-<Down arrow>.

To move back or forward a word at a time:
Use <Ctrl>-<Right arrow> and <Ctrl>-<Left arrow>.

To move up or down 8 lines at a time:
Use <Ctrl>-<Up arrow> and <Ctrl>-<Down arrow>.

As you insert blank lines, you may sometimes end up with blanks in both columns. You can delete them individually, but you may also wish to know how to delete a full line (BOTH columns).

To delete a WHOLE LINE:
Press <Shift>-<Ctrl>-<Enter>.

When you press Enter at the left margin of a column, everything in that column doown to a page break or a column break moves down one line. To insert a Column Break, press Alt-F6, F10, F8.

If you want to move text from one file to another, split the screen with F2, use the arrow keys to get into one window or the other, use F1 F6 to load the other file in that window, then press F6 to start highlighting, F6 again to select, F2 and an arrow key to move to the other window, F6 again to do the move. (F3 is the same but copies.)

12.7 Miscellaneous Notes

12.7.1 Segment Log File

The segment log file produced by the MAKELMK program is equivalent to the segment index file, but is in a form that you will find easier to read. Each segment number is shown on a separate line, followed by the special characteristics assigned to it -- ADDED, REMOVED, UNCHANGED, and whether it contains obsolete locations. A portion of the segment log for the 1980/1990 LKI conversion is shown below.

```
Segment 0324
Segment 0346    ADDED
Segment 0348                                        HAS OBSOLETE LOCATIONS
Segment 0351
Segment 0352                        UNCHANGED
Segment 0353                                        HAS OBSOLETE LOCATIONS
Segment 0354            REMOVED
Segment 0355
Segment 0356                        UNCHANGED
Segment 0357                        UNCHANGED
Segment 0359    ADDED
Segment 0361    ADDED
Segment 0362
```

12.7.2 Later Changes to Match Files

If you discover that some of the Landmark matching was incorrect, you can at any time correct them, recreate the Landmark Match List and the Segment Index files, upload them to the mainframe, and run Job PDSM again. Make sure you re-process **all** the match files, so that the match list and segment index files are complete.

It is also possible to modify the match list file on the mainframe directly, but if you do this, you will not be able to go back to the match files on the PC without losing the changes you made to the match list file.

12.7.3 Modify and Run THAS700

This section was written after the 1990 LKI conversion, and is included here unmodified. THAS700 was not used in the 1995 conversion.

The program THAS700 creates a file which contains the landmarks for each segment, in two columns. The first column contains the KMMARK and description of the landmarks in the older version of the LKI, and the second column contains the KMMARK and description of the landmarks in the newer version of the LKI. There is an example of a small portion of this file at the end of this section.

Check whether the latest and next-to-latest versions of the LKI are sorted by segment and kmmark. The 1990 LKI (TLKIP.LANDMARK) is; the 1980 LKI (TES.LANDMARK.DATA) was sorted by highway, segment, and kmmark. THAS700 needs its input files to be sorted by segment and kmmark only. If necessary, sort them into other files (don't change the master LKI!). See THASD.UTILJCL(SORTLMK) for an example; it sorts the 1980 LKI into a file called THASD.LANDMARK.OLD.

The JCL for THAS700 is in THASD.UTILJCL(RUN700). Change the DD statements for GO.OLDLMK, GO.NEWLMK, and GO.OLDNEW to refer to the appropriate files.

```
      GO.OLDLMK         -     The next-to-latest version of the LKI;
                                    currently THASD.LANDMARK.OLD
      GO.NEWLMK         -     The latest version of the LKI;
                                    currently TLKIP.LANDMARK
      GO.OLDNEW         -     The output file which will contain the old and
                              new landmarks for each segment;
                                    currently THASD.LANDMARK.OLDNEW
```

The source code for THAS700 is in THASD.SRCELIB(THAS700). The current version was written for the 1980 and 1990 LKIs, which are in very different formats. It is likely (but not certain) that future versions will be in the same format as the 1990 LKI. Currently, the old LKI has 80-character records, while the new LKI has 99-character records.

Change the following declarations in the source code as appropriate:

```
   OLDREC      -  Change the record length as needed.
   OLDRECTYPE  -  Probably not needed in future; replace by OLDREC#
                  (comparable to NEWREC#). Signalled whether a record was a
                  segment header (containing the segment number and
                  description) or an actual landmark record.
   OLDSEG      -  Segment number of a landmark in the old LKI.
   OLDKM       -  KMMARK of a landmark in the old LKI.
   OLDDESC     -  Description of a landmark in the old LKI.

   NEWREC      -  Change the record length as needed.
   NEWREC#     -  Some locations have many landmarks at one KMMARK. This
                  feature has been used to provide multi-line descriptions of
```

```
                  certain landmarks.  The first landmark of each such
                  grouping has NEWREC# = '1'.  Subsequent landmarks have
                  increasing values of NEWREC#.
   NEWSEG      -  Segment number of a landmark in the new LKI.
   NEWKM       -  KMMARK of a landmark in the new LKI.
   NEWDESC     -  Description of a landmark in the new LKI.
```

Change the DO UNTIL loops surrounding each READ statement as necessary. There are three such loops for each input file (old and new).

The DO UNTIL loops for the old file serve to skip the segment header lines, present in the 1980 LKI. If future releases of the LKI follow the 1990 format, the old file must be read in the same way the new file is read today.

The DO UNTIL loops for the new file eliminate all but the first line of multi-line descriptions (see NEWREC# above). Elke Pagel has stated a preference for having all the lines present in the Location-Code Transformation files; she finds them helpful in deciding which landmarks match which. Decide whether you wish to include these extra lines, and if so, eliminate the DO UNTIL loops
entirely. That is, change

```
      DO UNTIL (NEWREC# = '1' | $EOF2);
            READ FILE (NEWLMK) INTO (NEWREC);
      END;
```

to simply

```
      READ FILE (NEWLMK) INTO (NEWREC);
```

This should be done in the same way for both the old and the new landmark input files.

When the changes to THAS700 are complete, run it. From any PDF command line, type:

```
      TSO SUBMIT THASD.UTILJCL(RUN700)
```

or, from the TSO editor when editting the file, just type the command:

```
      SUBMIT
```

13 PLI Routine Index

Routine Naming Convention

The IBM system allows PLI routine names to be 7 characters long. Because PLI code of all Highways systems were originally kept in one large library (THWYP.SRCELIB), the first 4 characters of the names had to be used to identify the system. For this system it is THAS. Thus there are 3 characters left for the routine name.

E.g. routine 'abc' will be in member THASabc of library THASP.SRCELIB. If the name is numeric, the routine is a MAIN routine, if it is alphabetic, it is a subroutine.

Now that the HAS routines are in their own library (THASP.SRCELIB), all 7 characters could be used to create new routine names. Having all H.A.S, routines identified with the THAS prefix eliminates the chances of routine name collisions with other concatinated object or load libraries.

Main Routines

Update Sub-System

```
010 - copy data from the MVB file.
020 - extract fatal accident records.
030 - check location codes.
040 - remove Surrey jurisdiction 2 records.
050 - create the Edit file.
054 - effect the edit.
060 - split data by jurisdiction codes.
061 - set jurisdiction codes to 1.
071 - create the completion Job Log record.
075 - count records and accidents, and check for errors.
080 - extract key fields from an accident file (make CASELIST)
081 - reports duplicate case-date in a CASELIST file.
085 - Extract non-blank LocText data corresponding to an acc. file.
100 - remove old records from a (master) file.
101 - remove all accidents before given year from master accident file.
140 - calculate causal factors (weights)
141 - research version of THAS140.
142 - research version of THAS140.
```

Data Retrieval Sub-System

```
200 - Data Selection from the PDS Master Files.
203 - Select accidents from a subset, using accident data field criteria.
205 - Select by Counter-Measure Accident Type.
210 - Hazardous Locations.
220 - Hazardous Sections.
225 - Perform accident statistics for specified sections.
230 - Histogram Report.
232 - Create file with Fatal, Injury, PDO and total number of accidents.
235 - Counts Fatal/Inj/PDO accidents in increments.
240 - Details Report.
250 - Summary Report, version 1.
251 - Summary Report, version 2.
260 - Rate Table.
270 - calculate average accident type ratios
708 - Creates 4 empty files (SDATA, XDATA, SDESC, and XDESC).
710 - Create Victim File.
715 - Convert accident subset into a TOSS collision diagram file.
400 - Combined Hazardous Locations and Sections Report (SAS400).
```

SAS - Facility to run a SAS program on HAS accident subsets.

Utility Subsystem

105 - convert accident location codes from an old to a new LKI.
106 - 1996 version of THAS105.
110 - separate accident recs into segment and node files.
120 - prepare files for IEBGENER PDS Master creation.
132 - create the Segment-Highway-Node file.
134 - invert Segment-Node info in SEGMENT file, create Node-Segment file.
152 - do JAN90 transformation on Segment 0720
161 - Convert the locations in a SEGDIST file to a newer LKI.
162 - Convert the locations in a SEGCLASS file to a newer LKI.
163 - Convert the locations in the COUNTER.LOCNS file.
164 - Convert the COUNTER.MAP.SUBSEG.CSV file to a newer LKI.
700 - create old-new landmarks file.
708 - Creates 4 empty files (SDATA, XDATA, SDESC, and XDESC).
709 - Writes description files of data subset files created by SAS program.
710 - produces an output record for each victim in the victim table.
715 - Convert an accident subset into a TOSS collision diagram file.
720 - copy an accident file making specified data changes.
750 - make initial Counter Locations file.
752 - Create a comma-delimited volume file for downloading to a PC.
753 - Copy the Perm/Short Count system files into one file.
754 - Create a unique list of Traffic Counter Station ID's.
755 - Selects the station volume data required by the H.A.S.
756 - Create the Station Volumes file.
760 - create Segment Volumes file.
762 - produce the traffic volumes for the specified sub-segments.
764 - produce the traffic volumes for the specified nodes.
765 - Transforms the THAS300 RAWDATA file into the new SEGCLASS file.
766 - creates an output traffic volume file with FINE and COARSE data.
768 - Reads Node Volumes file ,fills in data for nodes not on that file.
770 - add HSNTAB sequence numbers to SEGCLASS file.
775 - add HSNTAB sequence numbers to SEGDIST file.
780 - determine a location code from the original Surrey code.
800 - Perform the RFI MODifications upon the landmark file.
802 - Remove 2 bytes of RFI ADD.TXT file;append ' (RFI)' to landmark description
804 - Set landmark numbers, and check KMs.
810 - Re-create the RFI Range Features File with date modifications.

Subroutines

On the following pages are two copies of an index of all the H.A.S. PLI subroutines. The first is sorted by the associated main program, and the second is sorted by Routine name.

The subroutine index tables are linked to Excel spreadsheet subroutines.xls. See the README worksheet in that spreadsheet for instructions on how to update the tables.

The three fields in the index are:

Main:

If a subroutine is associated with only one or two main routines, the main routine name(s) will be in the "Main" field. If a routine is used by many main routines, the "Main" field will contain one of the following:

- Util - a utility routine which could be useful in any system.

- Util-THAS - a utility routine which is (or could be) used throughout the H.A.S. but which is not likely to be useful in other systems.
- Util-TLKI - a utility routine created for the LKI system. Some of these routines are intended only for accessing LKI data; others are of general use.

Routine

In most cases, this is the 3 character routine name, for PL/I routines 'abc' in THASP.SRCELIB(THASabc). Some routines in the same library are written in SAS instead of PL/I; their full member names are given. That is, SAS routine SASabcd is in THASP.SRCELIB(SASabcd). The H.A.S. system also uses some LKI routines; these are shown as TLKIabc and can be found in THWYP.SRCELIB(TLKIabc).

Description

Many subroutines have long descriptive names which are used in the PLI code to make the code more readable. In this case the long name is given in upper case, usually at the start of the Description field. A one- or two-line description of the routine follows.

13.1 SUBROUTINES ORGANIZED BY MAIN PROGRAM

Main	Routine	Description
105	CBI	CHECK_FOR_BAD_INDEXES - checks Segment Index for bad indexes into Landmark Match List
105	CVK	CONVERT_KMMARK from old to new Landmark Match List ranges
105	FSI	FIND_SEGMENT_IN_INDEX - finds given seg. in Segment Index
105	HSC	HANDLE_SPECIAL_CASES - corrects location codes where there is an error in the current LKI
105	RML	READ_LANDMARK_MATCH_LIST - also reads Segment Index
105	SNX	SEARCH_SEGMENT_INDEX (from last find) for a given segment
105	TLC	TRANSLATE_LOCATION_CODE from old to new LKI
106	CV6	CONVERT_KMMARK - from OLD1 <= KMMARK <= OLD2 into corresponding KMMARK in range NEW1 <= KMMARK <= NEW2.
106	RM6	Reads landmark match lists and their associated segment indexes.
106	TL6	Translates location codes originally produced using an older LKI into corresponding location codes fitting a newer version of the LKI.
140	KNR	PROCEDURE (k, lbl,n, obs,nvar, W) REORDER - Calculate and return the label for OBS - three weights in W(3).Called from THAS140.
200	CHN	CHECK_NODE to see if its data should be included.
200	FAL	FILL_ALL - makes specific an 'ALL' From-To spec.
200	FSP	FILL_SPECS - fills in the blanks of a 'From-To' spec.
200	GDT	GET_DATA from the PDS Master, and write out.
200	GFL	GET_FIELD_CODES - read data field codes from SYSIN.
200	GFT	GET_FROM_TO_SPECS - read From-To specs from SYSIN.
200	GSP	GET_SPECS - reads control specs for THAS200
200	IN1	INTERSECT1 - forms the intersection of two individual FROM-TO specs (not FROM-TO lists)
200	IN2	INTERSECT2 - forms the intersection of two non-empty FROM-TO lists
200	IN3	INTERSECT3 - forms the intersection of three non-empty FROM-TO lists
200	INT	Creates a FROM-TO list that is the INTersection of all three input FROM-TO lists
200	MEM	LIST_MEMBERS to be read from the PDS Master.
200	OPP	For a list of From-To specs, generate and insert opposing sections.
200	PDS	calls the assembler routines which read the PDS's.
200	PFT	PUT_FROMTO - To WRITE A FROM-TO LIST TO FILE 'OUT'.
200	PSP	PUT_SPECS - writes out control specs
200	QUA	QUALIFIES - tests data fields for record inclusion.
200	RDN	Assembler routine to read from PDS with ddname NODPDS
200	RDS	Assembler routine to read from PDS with ddname SEGPDS
200	RSD	READ SECTION DEFINITIONS - Reads the seg and km information only, from a Section Definitions CSV file.
200	SCL	SELECT_CLASSES - generates a FROM-TO list of all highway sections with any of a set of given hwy. classifications
200	SDS	SELECT_DISTRICTS - generates a FROM-TO list of all highway sections in a given list of districts
210	CHA	CHECK_ACTIVE - decides if a location is hazardous.
210	CRA	CREATE_ACTIVE (potentially hazardous) location.
210	GS2	GET_SPECS - read control specs for THAS210
210	HLR	PRINT_REPORT - prints a Hazardous Location Report.
210	PS2	PUT_SPECS - writes out control specs

220	CAS	CHECK_ANY_SECTION - check section in SEC, containing accidents in buffer with record numbers from BR to ER, to see if it is hazardous.
220	CCS	CHECK_CURRENT_SECTION - is a section hazardous ?
220	EKM	calculates a seg-km FSLENGTH past a given seg-km.
220	FHS	FILL_HAZ_SECTION - calculates data for and fills one line of a hazardous section report record
220	GS3	GET_SPECS_3 - read control specs for THAS220
220	HSR	PRINT_REPORT - prints a Hazardous Section Report
220	PHS	PROCESS_HAZ_SECTION and write a report record.
220	PS3	PUT_SPECS - writes out control specs
220	V20	given the segment numbers of the ends of a section, determines a traffic volume for the section
230	BHP	BEGIN_HISTO_PAGE - prints histogram page headings
230	EHP	END_HISTO_PAGE - prints histogram scale line & page totals
230	FSN	FIND_SEGORNODE_IN_SEARCH_PATH - prints blank histogram (if needed) up to a given location in the search path
230	GUT	GET_USER_TITLE - reads the user-defined histogram title
230	HCS	READ_CSV_FILE_OPTIONS - Contains code for preparation of Histogram CSV (Comma-Separated-Values) output file.
230	HSP	GET_HISTOGRAM_SPECS - Reads the control parameters for a histogram from SYSIN.
230	MHD	MAKE_HISTO_DISPLAY - makes a string of F's, I's, and P's
230	MPC	MAKE_PAGE_CONTENTS - creates page-header strings
230	MTS	MAKE_TOTALS - creates totals string for a histogram
230	NDL	Determines the number of printed lines that will be required for the histogram display at the current location.
230	PHL	PRINT_HISTO_LINE - prints lines of a histogram
230	PMF	PRINT_MULTIPLIER_FACTORS - Prints header line describing how many accidents are represented by each F, I and P character.
230	PND	PRINT_NODE - prints the histogram for a node
230	PSG	PRINT_SEGMENT - prints the histogram for an entire segment
230	PST	PRINT_SEGMENT_TOTALS - prints seg. totals line & headings
230	RNS	REMOVE_NODES_FROM_SEGMENTS - modifies search path table so nodes are not included in KMMARK range of segments
230	RST	RESET_SEGMENT_TOTALS - zeroes the segment totals array
230	SBL	SUMMARIZE_ACCIDENTS_BY_LOCATION - creates histogram summary file
230	SPD	SET_SEARCH_PATH_DATA - sets variables corresponding to the current location in the search path
240	DR2	Convert accident record codes into text, and output to details report.
240	DR3	Convert vehicle codes to text, and output to details report.
240	DR4	Convert victim codes to text, and output to details report.
240	PR0	prints details report headings
240	PR1	prints accident descriptions (from THASC02) in 3 columns
240	PR2	prints vehicle descriptions (from THASC03)
240	PR3	prints victim descriptions (from THASC04)
250	BSP	BEGIN_SUMMARY_PAGE - prints ver.-1 summary report headings
251	BS1	BEGIN_SUMMARY_PAGE - prints ver.-2 summary report headings
251	PCD	PRINT_CODES - prints meanings of summary report codes
260	BRP	BEGIN_RATE_TABLE_PAGE - prints page and column headings
260	CAV	CALCULATE_ADJACENT_VOLUMES - calculates traffic vols. for one year from counters on seg(s) adjacent to current seg.
260	CEL	DETERMINE_CELLS of Rate Table containing a given location

260	COL	DETERMINE_COLUMNS - determines which columns of a Rate Table include a given highway class
260	COS	CHECK_OFF_SECTION in Highway Sections Table when the section is included in the Rate Table
260	CR6	CALC_CRITICAL_RATES_260 - calculates critical accident rates for the Rate Table program.
260	CRL	PRINT_CLASS_RATE_LINE to file SORTIN
260	CRR	PRINT_CLASS_RATE_REPORT - sorts and prints the report
260	CST	CALCULATE_STATS - calculates Rate Table statistics (mean and standard deviation)
260	DCR	DERIVE_CELL_RATES - calculates final accident & fatality rates for each cell
260	DSS	DIVIDE_SEGCLASS_SECTIONS - splits hwy sections in SEGCLASS table into smaller sections of approx. uniform length
260	FST	FILL_SECTION_TABLE_ENTRY - adds a section containing a given segment & KMMARK to the Highway Sections Table
260	GHS	GET_HSECTAB_INFO - given seg-km, returns section limits
260	IUS	INCLUDE_UNCHECKED_SECTIONS - includes highway sections without accidents in the Rate Table calculations
260	LAD	LOCNS_WITH_ACCIDENTS_DRIVER - driver routine for LOCATION-type Rate Table
260	LLD	LOCNS_AT_LANDMARKS_DRIVER - driver routine for LANDMARK-type Rate Table
260	PRT	PRINT_RATE_TABLE
260	RHO	REMOVE_HSECTAB_OVERLAPS
260	ROW	DETERMINE_ROWS - determines which rows of a Rate Table include a given daily traffic volume (ADT)
260	RTS	READ_RATE_TABLE_SPECS - reads SYSIN data
260	SCD	SECTIONS_DRIVER - driver routine for SECTION-type Rate Table
260	SHS	SUMMARIZE_HSECTAB - print # sections & total lengths
260	SPS	SPLIT_PATH_INTO_SECTIONS - builds the Highway Sections Table from the Search Path
260	TCL	TOTAL_CELL_LENGTHS - calculates total highway lengths in each cell
260	UAF	UPDATE_A_AND_F - adds # of accs. and deaths in the current section to each appropriate element of arrays A and FAT.
260	ULV	UPDATE_LV - adds L*V or M*V to each appropriate element of array LV
260	UST	UPDATE_STATRATES - saves accident rate for the current section in each appropriate element of array STATRATES
260	WPT	WHERE_IN_PATH - finds Search Path index of a location
270	PAT	Print_Accident_Type_key
270	R27	Print the program THAS270 Average Accident Type Ratios report
270	WAR	Write_Average_accident_type_Ratios
400	SASCDES	prints Combined Report description files (in SAS)
752	VBR	VOLUME_BREAKS - For the specified segment and year range, up to MAXN subsegments with constant traffic volume are returned in vectors KM1 and KM2.
760	AAA	AVERAGE_ADJACENT_ADTS - averages all ADTs from closest counters on 2 segs adjacent to current segment
760	AAS	AVERAGE_ADTS_ON_SEGMENT - averages all ADTs (annual and monthly) for all traffic counters on a segment
760	ASQ	ADJACENT_SEGMENT_QUALIFIES - decides if an adjacent segment qualifies for approximating volumes on current seg
760	PSC	PRINT_SEGMENT_COUNTERS - prints the stored report-line array for one segment

760	RAC	RECORD_ADJACENT_COUNTER - stores info about a counter adjacent to current segment in the report-line array
760	RSC	RECORD_SEGMENT_COUNTERS - stores info about all counters on a segment in the report-line array
768	CNV	CREATE NODE VOLUME - For the specified node and year, a traffic volume record (AADT and MADTS) is created averaging segment volume data for the segments named in the node name.
200 & 203	PFL	PUT_FIELD_CODES - To write the DATA FIELD CODE specs to file 'OUT'.
200 & 203	SFL	SELECT BY FIELD - To indicate whether an accident passes the selection criteria (for accident data field values.)
210 & 220	A41	PROCEDURE (ACCIDENT_RECORD, OBS_RECORD) REORDER - Create a 41 variable, 246-byte Diagnosis Accident 'OBServation' record from a page 1 accident record.
210 & 220	ACT	look up acc type number, return description.
210 & 220	CAR	CALCULATE_ACCIDENT_RATE
210 & 220	CCR	CALCULATE_CRITICAL_RATES
210 & 220	CMM	COUNTER_MEASURE_METHOD - calculate & check accident type ratios
210 & 220	DIS	GET_DISTANCE from the last discontinuity of a seg-km
210 & 220	EVC	EVALUATE_CRITERIA - to see if a loc or sec is hazardous.
210 & 220	HST	SORT_REPORT - assigns rank & sort for each report.
210 & 220	LSP	LOAD_SEARCH_PATH into memory from a description file.
210 & 220	NOD	returns the node name given a seg & kmmark at a node
210 & 220	PEC	PREPARE_AND_EVALUATE_CRITERIA
210 & 220	RCR	READ_CRITERIA - and interpret, for defining haz loc/sec.
210 & 220	SUC	SETUP_CRITERIA - entry in module EVC.
210 & 220	WDF	WRITE_DIAGNOSIS_FILE
210 & 220	WR2	checks flags and writes a record to two output files.
240 & 25x	TLU	TABLE_LOOKUP - finds meaning of a code from MV104 table
762 & 764	CSV	Combine seg volumes
SAS	DE2	writes description files corresponding to data subset files created by a SAS program.
SAS	OPN	creates 4 empty files (SDATA, XDATA, SDESC, and XDESC)
Util	ABT	ABorT - prints a message, sets return code, aborts.
Util	BLZ	Remove leading zeros from a string
Util	C03	given a segment number, returns segment name & length.
Util	CLI	Checks if a string is on a list - appends if not.
Util	CTR	CENTER_STRING - centers non-blank portion of input string
Util	CTY	ADD_CENTURY to a YYMMDD date to form a YYYYMMDD date.
Util	DAT	Returns the system Date and time as yy/mm/dd hh:mm
Util	DIF	Calculates difference in days between two yyyymmdd dates.
Util	DIM	returns the number of Days In the given Month
Util	DIR	loads member names of a PDS DIRectory into memory
Util	DOW	returns the DAY_OF_WEEK of a given date
Util	DPD	DBL_PERCENT_DIFF - determines % diff between 2 double-precision real numbers
Util	DSN	Given a DD name, returns the corresponding DS (dataset) name.
Util	DT8	Extracts a date from STR and returns a date in yyyymmdd format.
Util	EDS	EDIT_STRING - changes all occurrences of FROM to TO in a given fixed length character string.
Util	FDN	NFIELDS - Returns the number of delimited substrings
Util	IL2	INLIST2 - indicates if a pair of strings are in a list of pairs of strings.
Util	ILN	INLIST_NUMERIC - Indicates if an integer is in a list.
Util	INL	INLIST - Indicates if a string is in a list.
Util	JDC	Given a date in character form yyyymmdd, returns the julian day.

Util	JUL	returns a JULian day (# of days since 24 Nov. 4713 BC) given a Gregorian date (year, month, day)
Util	LJS	Left-Justifies a String (removes leading blanks).
Util	LYR	returns '1'B if given year is a LEAP_YEAR
Util	PDF	SGL_PERCENT_DIFF - determines % diff between 2 single-precision real numbers
Util	PDI	PARSE_DATE_INTEGERS - Separates a YYYYMMDD date into FIXED BIN(31) iyear (4 digit), imonth, and iday.
Util	PDT	PARSE_DATE - separates YYMMDD into year, month, day
Util	PIL	checks to see if string STR is in list LIST.
Util	RJS	right justifies a string
Util	RNG	(MONTH_)RANGE - determines which months, hours, etc. are included in a given range
Util	SRL	SEARCH_LIST - like INL, but also returns the pointer.
Util	SRT	SoRTs a file
Util	STI	Sets indeces in INDX pointing to elements in ARRIN in increasing order.
Util	TRM	Returns a STR with leading and trailing blanks removed
Util	UPC	translates a string to UPPER_CASE.
Util	VDT	checks a given date for ValiDiTy
Util	VWD	Breaks a character string up into blank-delimited words, returns them in an array of varying character strings.
Util	WDS	SEPARATE_WORDS - breaks a character string up into an array of blank_delineated words
Util	YMD	Converts a Julian day (number of days since Nov 24, 4713 BC) to a Gregorian date: Year, Month, Day.
Util-THAS	CCT	COMBINE COUNTER_TYPES - Returns a string containing a P if either CT1 or CT2 contains a P, and an S if either CT1 or CT2 contains an S.
Util-THAS	CIC	CLASS_IN_CLASS_SET - determines whether a given highway class is included in a given classification set
Util-THAS	CLM	CHECK_LANDMARK - is a locn at a LMK of listed types?
Util-THAS	CLN	FIND_CLASS_NAME - translates a highway classification code into its full name
Util-THAS	CRK	Convert Level of Significance to K, for Critical Acc. Rate calculations.
Util-THAS	CSD	COMBINE_SEGMENT_DATA - puts all SHNFIL & traffic volume data pertaining to one segment into a single structure
Util-THAS	DDF	DUMP_DESCRIPTION_FILE - prints INDESC file
Util-THAS	DES	Creates a DEScription file of a subset pair.
Util-THAS	DFD	extracts DESCRIPTION_FILE_DATES, months, and hours
Util-THAS	DMR	DAYS_IN_MONTH_IN_RANGE - Determines the number of days in the M of year Y, in the date range Y1/M1/D1 - Y2/M2/D2.
Util-THAS	DST	GET_DISTRICT name & number of a given highway location
Util-THAS	EOB	Extract 20 observation variables from the accident record in structure ACCSTRUC, for use in the fuzzy recognition algorithm. See main program THAS140 for details.
Util-THAS	FLD	look up a field name on file THASP.TABLE(FLDNAMES).
Util-THAS	FLN	Given a H.A.S. PL/I field name, look it up on the field names table, and return all the information from the table.
Util-THAS	GCR	GET_CRITICAL_RATE - looks up a highway class on the critical-rate table and returns a critical accident rate
Util-THAS	GDR	GET_DATE_RANGE from a description file
Util-THAS	GKD	GET_KMMARK_DESCRIPTION - finds landmark description (if any) for a given segment & KMMARK
Util-THAS	GLT	GET_LANDMARK_TYPE - lookup lmk type from seg, km
Util-THAS	GND	GET_NODE_DESCRIPTION - finds landmark description for a given node

Util-THAS GNL	GET_NEAREST_LANDMARK to a given location.
Util-THAS GRD	GET_REGION_DISTRICT - given seg & km, returns region & district numbers of that location
Util-THAS GRN	GET_REGION_DISTRICT_FOR_NODE - given a node name, returns a (consistent) region & district number for that node
Util-THAS GSC	GET_SEG_CLASS_INFO - given seg & km, returns boundaries & classification of hwy section containing that location.
Util-THAS GSK	GET_SEG_KM_FOR_NODE - finds a segment and KMMARK corresponding to a given node
Util-THAS HCM	HIGHWAY_CLASS_MATCH - checks if a highway class matches a highway class template.
Util-THAS HNL	separates highway letter & number
Util-THAS HWY	GET_HIGHWAY for a given segment from the HSN table.
Util-THAS LA2	Lookup Average Accident Rates from DDNAME AVERAT2
Util-THAS LAR	Lookup Average Accident Rates from DDNAME AVERAT
Util-THAS LCN	LOAD_CLASS_NAMES_TABLE - reads Class Names file
Util-THAS LCR	LOAD_CRITICAL_RATES - reads Critical Rates file
Util-THAS LLT	Lookup location text from a sequential LOCTEXT file in Case Date order
Util-THAS LLX	Lookup location text from the VSAM LOCTEXT file.
Util-THAS LMD	returns a landmark description, given a landmark code
Util-THAS LMK	LOOKUP_LANDMARK - finds a LOCN on the landmark file.
Util-THAS LMT	LANDMARK_TYPE - look up lmk type description from code.
Util-THAS LMV	Load the MV104 code table into memory
Util-THAS LSC	LOAD_SEG_CLASS_FILE into a table.
Util-THAS LSL	LOAD_SEGMENT_LANDMARKS - reads all landmarks for a segment
Util-THAS LSR	LOAD_SEGMENT_RANGE_FEATURES for one segment.
Util-THAS LSW	LOAD_SWAR_WEIGHTS - from DD name SWARWTS.
Util-THAS LUC	searches a CASELIST file for the provided Case and Date
Util-THAS MES	Prints identifying fields of an acc. rec. and a message.
Util-THAS MLI	MAKE_LOCN_ID - constructs a location ID
Util-THAS NCS	NORMALIZE_CLASS_SET - orders characterstic codes as in the Class Names file.
Util-THAS NT1	Translates 1st char of a string from digit to letter.
Util-THAS NT2	Reverse of NT1 (member name to node name).
Util-THAS NVL	NODE_VOLUME - determines traffic volume at a node
Util-THAS OP1	Generate opposing start and end points from given starting end points.
Util-THAS PCR	PUT_CRITICAL_RATES - writes critical rates table
Util-THAS PRC	PRints record and accident Counts.
Util-THAS RDA	READ_ACCIDENTS - Reads all the records of one accident.
Util-THAS REL	determines the RELATIVE_POSITION of two locations
Util-THAS RGF	LOOKUP_RANGE_FEATURE - in the Range Feature Table, given Segment and Kmmark.
Util-THAS RNM	Reads a record from, the Node Counter Map CSV file defined in MAPFILE.
Util-THAS RP1	READ_PAGE1_ACCIDENT_ON_PATH - skips non-page-1 accs. and node accidents coded on segments not on search path
Util-THAS RPA	READ_PAGE1_ACCIDENT - skips non-page-1 accident records
Util-THAS RPL	READ_PAGE1_ACCIDENT_TILL_LOCN - like RP1, but also skips accidents before a given location
Util-THAS RSM	Reads a record from, the Subsegment Counter Map CSV file defined in MAPFILE
Util-THAS SASDESC	prints an accident subset description file (in SAS)
Util-THAS SAT	SET_ACCIDENT_TYPE_flags
Util-THAS SBN	SEGMENT_BEGIN_NODE - returns the begin node of a segment.

Util-THAS	SCS	SECTION_SEGMENT_CLASSIFICATION - returns the Highway Classification of a highway section.
Util-THAS	SDH	READ_SECTION_DEFINITIION_HEADER - Open file SDFILE, read the Section Definition file header, and set pointers to the fields.
Util-THAS	SDR	Read Section Definition Record
Util-THAS	SDT	LOAD_SEGDIST_TABLE - reads Segment-Region-District table from SEGDIST file
Util-THAS	SEN	SEGMENT_END_NODE - returns the end node of a segment.
Util-THAS	SIH	SEGMENT_IN_HIGHWAY - given hwy & seg, returns true if hwy is any of the up to 3 highways of the segment.
Util-THAS	SNT	Reads the highway-Segment-Node Table into memory
Util-THAS	SRY	Receives a Surrey MVB location code, looks it up on the SRYCONV table, and if found, returns the corresponding LKI location code in LKI.
Util-THAS	STV	READ_STATION_VOLUMES - reads all station volumes for one counter into a combined segment data structure
Util-THAS	SUR	Receives a Surrey MVB location code, looks it up on the SRY2LKI table, and if found, returns the corresponding LKI location code in LKI.
Util-THAS	SVI	SEGMENT_VOLUMES_INDEX - finds index into Segment Volumes Table for a given segment and year
Util-THAS	SVT	LOAD_SEGMENT_VOLUMES_TABLE - reads Segment Volumes file
Util-THAS	T01	Finds the first segment of a highway in the hSN Table.
Util-THAS	T02	Finds the last segment of a highway in the hSN Table.
Util-THAS	T03	Finds a segment in the highway-Segment-Node Table.
Util-THAS	T04	Looks up the length of a given segment.
Util-THAS	TCS	TRANSLATE_CLASSIFICATION_SET into its full English description
Util-THAS	TL1	TRANSLATE_1_LOCATION_CODE - Translates a location code according to the SEGMENT INDEX and LANDMARK.MATCH.LIST read from files SINDEX and LMATCH.
Util-THAS	TSL	Calculate total length of search path, for each year in specified year range.
Util-THAS	URC	Returns the category number of the first category in the Class Names table which contains a characteristic of "Urban".
Util-THAS	VLN	NODE_TRAFFIC_VOLUME - Determines total traffic volume at a node for a specific time period.
Util-THAS	VLS	SECTION_TRAFFIC_VOLUME - For the specified time period and section of highway, the total traffic volume (# vehicles) and the average daily traffic volume (in vehicles per day) are calculated.
Util-THAS	VOL	determines total traffic VOLume on a segment for a specific time period
Util-THAS	VSS	SUBSEGMENT_TRAFFIC_VOLUME - For the specified time period and sub-segment, the total traffic volume (# vehicles) and the daily traffic volume (vehicles per day) are calculated.
Util-THAS	WCS	Creates and writes a CSV record, using external data in THASXAFC.
Util-THAS	WRA	WRITE_ACCIDENT - Writes all the records of one accident
Util-THAS	ZKM	zero fills a KMMARK field
Util-TLKI	A05	given a segment number, returns district(s) it belongs to. (TLKIA05 is a separate ENTRY within TLKIC05.)
Util-TLKI	A06	given a segment #, returns highway(s) it belongs to. (TLKIA06 is a separate ENTRY within TLKIC06.)
Util-TLKI	A07	given segment #, returns RCMP detachment(s) it belongs to. (TLKIA07 is a separate ENTRY within TLKIC07.)
Util-TLKI	BRK	finds a word-boundary break point within a string.
Util-TLKI	C01	given a district #, returns district name & region number.
Util-TLKI	C02	given a highway number, returns the highway name.
Util-TLKI	C04	given an RCMP detachment number, returns detachment name.
Util-TLKI	C05	given seg.# & district #, returns start & end km & offset.

Util-TLKI	C07	given a segment # & RCMP detachment #, returns attribute fields.
Util-TLKI	C08	given a region #, returns the region name.
Util-TLKI	DTR	converts a date in YYMMDD format to DD MMM YY format.
Util-TLKI	LEN	determines non-blank length of a character string.

13.2 SUBROUTINES SORTED BY ROUTINE NAME

Main	Routine	Description
Util-TLKI	A05	given a segment number, returns district(s) it belongs to. (TLKIA05 is a separate ENTRY within TLKIC05.)
Util-TLKI	A06	given a segment #, returns highway(s) it belongs to. (TLKIA06 is a separate ENTRY within TLKIC06.)
Util-TLKI	A07	given segment #, returns RCMP detachment(s) it belongs to. (TLKIA07 is a separate ENTRY within TLKIC07.)
210 & 220	A41	PROCEDURE (ACCIDENT_RECORD, OBS_RECORD) REORDER - Create a 41 variable, 246-byte Diagnosis Accident 'OBServation' record from a page 1 accident record.
760	AAA	AVERAGE_ADJACENT_ADTS - averages all ADTs from closest counters on 2 segs adjacent to current segment
760	AAS	AVERAGE_ADTS_ON_SEGMENT - averages all ADTs (annual and monthly) for all traffic counters on a segment
Util	ABT	ABorT - prints a message, sets return code, aborts.
210 & 220	ACR	Analyse and check SYSIN criteria for 210 and 220
210 & 220	ACT	look up acc type number, return description.
Util	AFC	ADDFIELD_CHAR - for creating CSV output
760	ASQ	ADJACENT_SEGMENT_QUALIFIES - decides if an adjacent segment qualifies for approximating volumes on current seg
210 & 220	ATR	LOOKUP_AVERAGE_ACCIDENT_TYPE_RATIOS
230	BHP	BEGIN_HISTO_PAGE - prints histogram page headings
Util	BLZ	Remove leading zeros from a string
Util-TLKI	BRK	finds a word-boundary break point within a string.
260	BRP	BEGIN_RATE_TABLE_PAGE - prints page and column headings
251	BS1	BEGIN_SUMMARY_PAGE - prints ver.-2 summary report headings
250	BSP	BEGIN_SUMMARY_PAGE - prints ver.-1 summary report headings
Util-TLKI	C01	given a district #, returns district name & region number.
Util-TLKI	C02	given a highway number, returns the highway name.
Util	C03	given a segment number, returns segment name & length.
Util-TLKI	C04	given an RCMP detachment number, returns detachment name.
Util-TLKI	C05	given seg.# & district #, returns start & end km & offset.
Util-TLKI	C07	given a segment # & RCMP detachment #, returns attribute fields.
Util-TLKI	C08	given a region #, returns the region name.
210 & 220	CAR	CALCULATE_ACCIDENT_RATE
220	CAS	CHECK_ANY_SECTION - check section in SEC, containing accidents in buffer with record numbers from BR to ER, to see if it is hazardous.
260	CAV	CALCULATE_ADJACENT_VOLUMES - calculates traffic vols. for one year from counters on seg(s) adjacent to current seg.
105	CBI	CHECK_FOR_BAD_INDEXES - checks Segment Index for bad indexes into Landmark Match List
210 & 220	CCR	CALCULATE_CRITICAL_RATES
220	CCS	CHECK_CURRENT_SECTION - is a section hazardous ?
Util-THAS	CCT	COMBINE COUNTER_TYPES - Returns a string containing a P if either CT1 or CT2 contains a P, and an S if either CT1 or CT2 contains an S.
260	CEL	DETERMINE_CELLS of Rate Table containing a given location
210	CHA	CHECK_ACTIVE - decides if a location is hazardous.
200	CHN	CHECK_NODE to see if its data should be included.
Util-THAS	CIC	CLASS_IN_CLASS_SET - determines whether a given highway class is included in a given classification set

Util	CLI	Checks if a string is on a list - appends if not.
Util-THAS	CLM	CHECK_LANDMARK - is a locn at a LMK of listed types?
Util-THAS	CLN	FIND_CLASS_NAME - translates a highway classification code into its full name
210 & 220	CMM	COUNTER_MEASURE_METHOD - calculate & check accident type ratios
768	CNV	CREATE NODE VOLUME - For the specified node and year, a traffic volume record (AADT and MADTS) is created averaging segment volume data for the segments named in the node name.
260	COL	DETERMINE_COLUMNS - determines which columns of a Rate Table include a given highway class
260	COS	CHECK_OFF_SECTION in Highway Sections Table when the section is included in the Rate Table
260	CR6	CALC_CRITICAL_RATES_260 - calculates critical accident rates for the Rate Table program.
210	CRA	CREATE_ACTIVE (potentially hazardous) location.
Util-THAS	CRK	Convert Level of Significance to K, for Critical Acc. Rate calculations.
260	CRL	PRINT_CLASS_RATE_LINE to file SORTIN
260	CRR	PRINT_CLASS_RATE_REPORT - sorts and prints the report
Util-THAS	CSD	COMBINE_SEGMENT_DATA - puts all SHNFIL & traffic volume data pertaining to one segment into a single structure
260	CST	CALCULATE_STATS - calculates Rate Table statistics (mean and standard deviation)
762 & 764	CSV	Combine seg volumes
Util	CTR	CENTER_STRING - centers non-blank portion of input string
Util	CTY	ADD_CENTURY to a YYMMDD date to form a YYYYMMDD date.
106	CV6	CONVERT_KMMARK - from OLD1 <= KMMARK <= OLD2 into corresponding KMMARK in range NEW1 <= KMMARK <= NEW2.
105	CVK	CONVERT_KMMARK from old to new Landmark Match List ranges
Util	DAT	Returns the system Date and time as yy/mm/dd hh:mm
260	DCR	DERIVE_CELL_RATES - calculates final accident & fatality rates for each cell
Util-THAS	DDF	DUMP_DESCRIPTION_FILE - prints INDESC file
SAS	DE2	writes description files corresponding to data subset files created by a SAS program.
Util-THAS	DES	Creates a DEScription file of a subset pair.
Util-THAS	DFD	extracts DESCRIPTION_FILE_DATES, months, and hours
Util	DIF	Calculates difference in days between two yyyymmdd dates.
Util	DIM	returns the number of Days In the given Month
Util	DIR	loads member names of a PDS DIRectory into memory
210 & 220	DIS	GET_DISTANCE from the last discontinuity of a seg-km
Util-THAS	DLM	Gets dates from DATELIMS file.
Util-THAS	DMR	DAYS_IN_MONTH_IN_RANGE - Determines the number of days in the M of year Y, in the date range Y1/M1/D1 - Y2/M2/D2.
Util	DOW	returns the DAY_OF_WEEK of a given date
Util	DPD	DBL_PERCENT_DIFF - determines % diff between 2 double-precision real numbers
240	DR2	Convert accident record codes into text, and output to details report.
240	DR3	Convert vehicle codes to text, and output to details report.
240	DR4	Convert victim codes to text, and output to details report.
Util	DSN	Given a DD name, returns the corresponding DS (dataset) name.
260	DSS	DIVIDE_SEGCLASS_SECTIONS - splits hwy sections in SEGCLASS table into smaller sections of approx. uniform length
Util-THAS	DST	GET_DISTRICT name & number of a given highway location
Util	DT8	Extracts a date from STR and returns a date in yyyymmdd format.

Util-TLKI	DTR	converts a date in YYMMDD format to DD MMM YY format.
Util	EDS	EDIT_STRING - changes all occurrences of FROM to TO in a given fixed length character string.
230	EHP	END_HISTO_PAGE - prints histogram scale line & page totals
220	EKM	calculates a seg-km FSLENGTH past a given seg-km.
Util-THAS	EOB	Extract 20 observation variables from the accident record in structure ACCSTRUC, for use in the fuzzy recognition algorithm. See main program THAS140 for details.
210 & 220	EVC	EVALUATE_CRITERIA - to see if a loc or sec is hazardous.
200	FAL	FILL_ALL - makes specific an 'ALL' From-To spec.
Util	FDN	NFIELDS - Returns the number of delimited substrings
220	FHS	FILL_HAZ_SECTION - calculates data for and fills one line of a hazardous section report record
Util-THAS	FLD	look up a field name on file THASP.TABLE(FLDNAMES).
Util-THAS	FLN	Given a H.A.S. PL/I field name, look it up on the field names table, and return all the information from the table.
105	FSI	FIND_SEGMENT_IN_INDEX - finds given seg. in Segment Index
230	FSN	FIND_SEGORNODE_IN_SEARCH_PATH - prints blank histogram (if needed) up to a given location in the search path
200	FSP	FILL_SPECS - fills in the blanks of a 'From-To' spec.
260	FST	FILL_SECTION_TABLE_ENTRY - adds a section containing a given segment & KMMARK to the Highway Sections Table
Util-THAS	GCR	GET_CRITICAL_RATE - looks up a highway class on the critical-rate table and returns a critical accident rate
Util-THAS	GDR	GET_DATE_RANGE from a description file
200	GDT	GET_DATA from the PDS Master, and write out.
200	GFL	GET_FIELD_CODES - read data field codes from SYSIN.
200	GFT	GET_FROM_TO_SPECS - read From-To specs from SYSIN.
260	GHS	GET_HSECTAB_INFO - given seg-km, returns section limits
Util-THAS	GKD	GET_KMMARK_DESCRIPTION - finds landmark description (if any) for a given segment & KMMARK
Util-THAS	GLT	GET_LANDMARK_TYPE - lookup lmk type from seg, km
Util-THAS	GND	GET_NODE_DESCRIPTION - finds landmark description for a given node
Util-THAS	GNL	GET_NEAREST_LANDMARK to a given location.
Util-THAS	GRD	GET_REGION_DISTRICT - given seg & km, returns region & district numbers of that location
Util-THAS	GRN	GET_REGION_DISTRICT_FOR_NODE - given a node name, returns a (consistent) region & district number for that node
210	GS2	GET_SPECS - read control specs for THAS210
220	GS3	GET_SPECS_3 - read control specs for THAS220
Util-THAS	GSC	GET_SEG_CLASS_INFO - given seg & km, returns boundaries & classification of hwy section containing that location.
Util-THAS	GSK	GET_SEG_KM_FOR_NODE - finds a segment and KMMARK corresponding to a given node
200	GSP	GET_SPECS - reads control specs for THAS200
230	GUT	GET_USER_TITLE - reads the user-defined histogram title
Util-THAS	HCM	HIGHWAY_CLASS_MATCH - checks if a highway class matches a highway class template.
230	HCS	READ_CSV_FILE_OPTIONS - Contains code for preparation of Histogram CSV (Comma-Separated-Values) output file.
210	HLR	PRINT_REPORT - prints a Hazardous Location Report.
Util-THAS	HNL	separates highway letter & number
105	HSC	HANDLE_SPECIAL_CASES - corrects location codes where there is an error in the current LKI

230	HSP	GET_HISTOGRAM_SPECS - Reads the control parameters for a histogram from SYSIN.
220	HSR	PRINT_REPORT - prints a Hazardous Section Report
210 & 220	HST	SORT_REPORT - assigns rank & sort for each report.
Util-THAS	HWY	GET_HIGHWAY for a given segment from the HSN table.
Util	IL2	INLIST2 - indicates if a pair of strings are in a list of pairs of strings.
Util	ILN	INLIST_NUMERIC - Indicates if an integer is in a list.
200	IN1	INTERSECT1 - forms the intersection of two individual FROM-TO specs (not FROM-TO lists)
200	IN2	INTERSECT2 - forms the intersection of two non-empty FROM-TO lists
200	IN3	INTERSECT3 - forms the intersection of three non-empty FROM-TO lists
Util	INL	INLIST - Indicates if a string is in a list.
200	INT	Creates a FROM-TO list that is the INTersection of all three input FROM-TO lists
260	IUS	INCLUDE_UNCHECKED_SECTIONS - includes highway sections without accidents in the Rate Table calculations
Util	JDC	Given a date in character form yyyymmdd, returns the julian day.
Util	JUL	returns a JULian day (# of days since 24 Nov. 4713 BC) given a Gregorian date (year, month, day)
140	KNR	PROCEDURE (k, lbl,n, obs,nvar, W) REORDER - Calculate and return the label for OBS - three weights in W(3).Called from THAS140.
260	LAD	LOCNS_WITH_ACCIDENTS_DRIVER - driver routine for LOCATION-type Rate Table
Util-THAS	LAR	Lookup Average Accident Rates from DDNAME AVERAT
Util-THAS	LA2	Lookup Average Accident Rates from DDNAME AVERAT2
Util-THAS	LCN	LOAD_CLASS_NAMES_TABLE - reads Class Names file
Util-THAS	LCR	LOAD_CRITICAL_RATES - reads Critical Rates file
Util-TLKI	LEN	determines non-blank length of a character string.
Util	LJS	Left-Justifies a String (removes leading blanks).
260	LLD	LOCNS_AT_LANDMARKS_DRIVER - driver routine for LANDMARK-type Rate Table
Util-THAS	LLT	Lookup location text from a sequential LOCTEXT file in Case Date order
Util-THAS	LLX	Lookup location text from the VSAM LOCTEXT file.
Util-THAS	LMD	returns a landmark description, given a landmark code
Util-THAS	LMK	LOOKUP_LANDMARK - finds a LOCN on the landmark file.
Util-THAS	LMT	LANDMARK_TYPE - look up lmk type description from code.
Util-THAS	LMV	Load the MV104 code table into memory
Util-THAS	LSC	LOAD_SEG_CLASS_FILE into a table.
Util-THAS	LSL	LOAD_SEGMENT_LANDMARKS - reads all landmarks for a segment
210 & 220	LSP	LOAD_SEARCH_PATH into memory from a description file.
Util-THAS	LSR	LOAD_SEGMENT_RANGE_FEATURES for one segment.
Util-THAS	LSW	LOAD_SWAR_WEIGHTS - from DD name SWARWTS.
Util-THAS	LUC	searches a CASELIST file for the provided Case and Date
Util	LYR	returns '1'B if given year is a LEAP_YEAR
200	MEM	LIST_MEMBERS to be read from the PDS Master.
Util-THAS	MES	Prints identifying fields of an acc. rec. and a message.
230	MHD	MAKE_HISTO_DISPLAY - makes a string of F's, I's, and P's
Util-THAS	MLI	MAKE_LOCN_ID - constructs a location ID
230	MPC	MAKE_PAGE_CONTENTS - creates page-header strings
230	MTS	MAKE_TOTALS - creates totals string for a histogram
Util-THAS	NCS	NORMALIZE_CLASS_SET - orders characterstic codes as in the Class Names file.
230	NDL	Determines the number of printed lines that will be required for the histogram display at the current location.

210 & 220	NOD	returns the node name given a seg & kmmark at a node
Util-THAS	NT1	Translates 1st char of a string from digit to letter.
Util-THAS	NT2	Reverse of NT1 (member name to node name).
Util-THAS	NVL	NODE_VOLUME - determines traffic volume at a node
200	OPP	For a list of From-To specs, generate and insert opposing sections.
Util-THAS	OP1	Generate opposing start and end points from given starting end points.
SAS	OPN	creates 4 empty files (SDATA, XDATA, SDESC, and XDESC)
270	PAT	Print_Accident_Type_key
251	PCD	PRINT_CODES - prints meanings of summary report codes
Util-THAS	PCR	PUT_CRITICAL_RATES - writes critical rates table
Util	PDF	SGL_PERCENT_DIFF - determines % diff between 2 single-precision real numbers
Util	PDI	PARSE_DATE_INTEGERS - Separates a YYYYMMDD date into FIXED BIN(31) iyear (4 digit), imonth, and iday.
200	PDS	calls the assembler routines which read the PDS's.
Util	PDT	PARSE_DATE - separates YYMMDD into year, month, day
210 & 220	PEC	PREPARE_AND_EVALUATE_CRITERIA
200 & 203	PFL	PUT_FIELD_CODES - To write the DATA FIELD CODE specs to file 'OUT'.
200	PFT	PUT_FROMTO - To WRITE A FROM-TO LIST TO FILE 'OUT'.
230	PHL	PRINT_HISTO_LINE - prints lines of a histogram
220	PHS	PROCESS_HAZ_SECTION and write a report record.
Util	PIL	checks to see if string STR is in list LIST.
230	PMF	PRINT_MULTIPLIER_FACTORS - Prints header line describing how many accidents are represented by each F, I and P character.
230	PND	PRINT_NODE - prints the histogram for a node
240	PR0	prints details report headings
240	PR1	prints accident descriptions (from THASC02) in 3 columns
240	PR2	prints vehicle descriptions (from THASC03)
240	PR3	prints victim descriptions (from THASC04)
Util-THAS	PRC	PRints record and accident Counts.
260	PRT	PRINT_RATE_TABLE
210	PS2	PUT_SPECS - writes out control specs
220	PS3	PUT_SPECS - writes out control specs
760	PSC	PRINT_SEGMENT_COUNTERS - prints the stored report-line array for one segment
230	PSG	PRINT_SEGMENT - prints the histogram for an entire segment
200	PSP	PUT_SPECS - writes out control specs
230	PST	PRINT_SEGMENT_TOTALS - prints seg. totals line & headings
200	QUA	QUALIFIES - tests data fields for record inclusion.
270	R27	Print the program THAS270 Average Accident Type Ratios report
760	RAC	RECORD_ADJACENT_COUNTER - stores info about a counter adjacent to current segment in the report-line array
210 & 220	RCR	READ_CRITERIA - and interpret, for defining haz loc/sec.
Util-THAS	RDA	READ_ACCIDENTS - Reads all the records of one accident.
200	RDN	Assembler routine to read from PDS with ddname NODPDS
200	RDS	Assembler routine to read from PDS with ddname SEGPDS
Util-THAS	REL	determines the RELATIVE_POSITION of two locations
Util-THAS	RGF	LOOKUP_RANGE_FEATURE - in the Range Feature Table, given Segment and Kmmark.
260	RHO	REMOVE_HSECTAB_OVERLAPS
Util	RJS	right justifies a string
106	RM6	Reads landmark match lists and their associated segment indexes.
105	RML	READ_LANDMARK_MATCH_LIST - also reads Segment Index

Util	RNG	(MONTH_)RANGE - determines which months, hours, etc. are included in a given range
Util-THAS	RNM	Reads a record from, the Node Counter Map CSV file defined in MAPFILE.
230	RNS	REMOVE_NODES_FROM_SEGMENTS - modifies search path table so nodes are not included in KMMARK range of segments
260	ROW	DETERMINE_ROWS - determines which rows of a Rate Table include a given daily traffic volume (ADT)
Util-THAS	RP1	READ_PAGE1_ACCIDENT_ON_PATH - skips non-page-1 accs. and node accidents coded on segments not on search path
Util-THAS	RPA	READ_PAGE1_ACCIDENT - skips non-page-1 accident records
Util-THAS	RPL	READ_PAGE1_ACCIDENT_TILL_LOCN - like RP1, but also skips accidents before a given location
760	RSC	RECORD_SEGMENT_COUNTERS - stores info about all counters on a segment in the report-line array
200	RSD	READ SECTION DEFINITIONS - Reads the seg and km information only, from a Section Definitions CSV file.
Util-THAS	RSM	Reads a record from, the Subsegment Counter Map CSV file defined in MAPFILE
230	RST	RESET_SEGMENT_TOTALS - zeroes the segment totals array
260	RTS	READ_RATE_TABLE_SPECS - reads SYSIN data
400	SASCDES	prints Combined Report description files (in SAS)
Util-THAS	SASDESC	prints an accident subset description file (in SAS)
Util-THAS	SAT	SET_ACCIDENT_TYPE_flags
230	SBL	SUMMARIZE_ACCIDENTS_BY_LOCATION - creates histogram summary file
Util-THAS	SBN	SEGMENT_BEGIN_NODE - returns the begin node of a segment.
260	SCD	SECTIONS_DRIVER - driver routine for SECTION-type Rate Table
Util-THAS	SCS	SECTION_SEGMENT_CLASSIFICATION - returns the Highway Classification of a highway section.
200	SCL	SELECT_CLASSES - generates a FROM-TO list of all highway sections with any of a set of given hwy. classifications
Util-THAS	SDH	READ_SECTION_DEFINITIION_HEADER - Open file SDFILE, read the Section Definition file header, and set pointers to the fields.
Util-THAS	SDR	Read Section Definition Record
200	SDS	SELECT_DISTRICTS - generates a FROM-TO list of all highway sections in a given list of districts
Util-THAS	SDT	LOAD_SEGDIST_TABLE - reads Segment-Region-District table from SEGDIST file
Util-THAS	SEN	SEGMENT_END_NODE - returns the end node of a segment.
200 & 203	SFL	SELECT BY FIELD - To indicate whether an accident passes the selection criteria (for accident data field values.)
260	SHS	SUMMARIZE_HSECTAB - print # sections & total lengths
Util-THAS	SIH	SEGMENT_IN_HIGHWAY - given hwy & seg, returns true if hwy is any of the up to 3 highways of the segment.
Util-THAS	SNT	Reads the highway-Segment-Node Table into memory
105	SNX	SEARCH_SEGMENT_INDEX (from last find) for a given segment
230	SPD	SET_SEARCH_PATH_DATA - sets variables corresponding to the current location in the search path
260	SPS	SPLIT_PATH_INTO_SECTIONS - builds the Highway Sections Table from the Search Path
Util	SRL	SEARCH_LIST - like INL, but also returns the pointer.
Util	SRT	SoRTs a file
Util-THAS	SRY	Receives a Surrey MVB location code, looks it up on the SRYCONV table, and if found, returns the corresponding LKI location code in LKI.

Util	STC	Char version of STI
Util	STI	Sets indeces in INDX pointing to elements in ARRIN in increasing order.
Util-THAS	STV	READ_STATION_VOLUMES - reads all station volumes for one counter into a combined segment data structure
210 & 220	SUC	SETUP_CRITERIA - entry in module EVC.
Util-THAS	SUR	Receives a Surrey MVB location code, looks it up on the SRY2LKI table, and if found, returns the corresponding LKI location code in LKI.
Util-THAS	SVI	SEGMENT_VOLUMES_INDEX - finds index into Segment Volumes Table for a given segment and year
Util-THAS	SVT	LOAD_SEGMENT_VOLUMES_TABLE - reads Segment Volumes file
Util-THAS	T01	Finds the first segment of a highway in the hSN Table.
Util-THAS	T02	Finds the last segment of a highway in the hSN Table.
Util-THAS	T03	Finds a segment in the highway-Segment-Node Table.
Util-THAS	T04	Looks up the length of a given segment.
260	TCL	TOTAL_CELL_LENGTHS - calculates total highway lengths in each cell
Util-THAS	TCS	TRANSLATE_CLASSIFICATION_SET into its full English description
Util-THAS	TL1	TRANSLATE_1_LOCATION_CODE - Translates a location code according to the SEGMENT INDEX and LANDMARK.MATCH.LIST read from files SINDEX and LMATCH.
106	TL6	Translates location codes originally produced using an older LKI into corresponding location codes fitting a newer version of the LKI.
105	TLC	TRANSLATE_LOCATION_CODE from old to new LKI
240 & 25x	TLU	TABLE_LOOKUP - finds meaning of a code from MV104 table
Util	TRM	Returns a STR with leading and trailing blanks removed
Util-THAS	TSL	Calculate total length of search path, for each year in specified year range.
260	UAF	UPDATE_A_AND_F - adds # of accs. and deaths in the current section to each appropriate element of arrays A and FAT.
260	ULV	UPDATE_LV - adds L*V or M*V to each appropriate element of array LV
Util	UPC	translates a string to UPPER_CASE.
Util-THAS	URC	Returns the category number of the first category in the Class Names table which contains a characteristic of "Urban".
260	UST	UPDATE_STATRATES - saves accident rate for the current section in each appropriate element of array STATRATES
220	V20	given the segment numbers of the ends of a section, determines a traffic volume for the section
752	VBR	VOLUME_BREAKS - For the specified segment and year range, up to MAXN subsegments with constant traffic volume are returned in vectors KM1 and KM2.
Util	VDT	checks a given date for ValiDiTy
Util-THAS	VLN	NODE_TRAFFIC_VOLUME - Determines total traffic volume at a node for a specific time period.
Util-THAS	VLS	SECTION_TRAFFIC_VOLUME - For the specified time period and section of highway, the total traffic volume (# vehicles) and the average daily traffic volume (in vehicles per day) are calculated.
Util-THAS	VOL	determines total traffic VOLume on a segment for a specific time period
Util-THAS	VSS	SUBSEGMENT_TRAFFIC_VOLUME - For the specified time period and sub-segment, the total traffic volume (# vehicles) and the daily traffic volume (vehicles per day) are calculated.
Util	VWD	Breaks a character string up into blank-delimited words, returns them in an array of varying character strings.
270	WAR	Write_Average_accident_type_Ratios
Util-THAS	WCS	Creates and writes a CSV record, using external data in THASXAFC.
210 & 220	WDF	WRITE_DIAGNOSIS_FILE

Util	WDS	SEPARATE_WORDS - breaks a character string up into an array of blank_delineated words
260	WPT	WHERE_IN_PATH - finds Search Path index of a location
210 & 220	WR2	checks flags and writes a record to two output files.
Util-THAS	WRA	WRITE_ACCIDENT - Writes all the records of one accident
Util	YMD	Converts a Julian day (number of days since Nov 24, 4713 BC) to a Gregorian date: Year, Month, Day.
Util-THAS	ZKM	zero fills a KMMARK field