INFORMATION RETRIEVAL SYSTEM

SYSTEM OS/360

VERSION 1.0
RUN WITH CURRENT DATA BASE 2
DATE 01/12/76
TIME 01:40:15
COMMAND:

RETRIEVE THE TEXT OF THE 3 POEMS WITH THE FOLLOWING CHARACTERISTICS:
SEA & BLOOD | SEA &-COLD| (HEART&COLD);

THE TITLES AND TEXTS ARE:

BYZANTIUM

THE UNPURGED IMAGES OF DAY RECEDE;
THE EMPEROR'S DRUNKEN SOLDIERY ARE ABED;
NIGHT RESONANCE RECEDES, NIGHT-WALKERS' SONG
AFTER GREAT CATHEDRAL GONG;
A STARLIT OR A MOONLIT DOME DISDAINS
ALL THAT MAN IS,
ALL MERE COMPLEXITIES,
THE FURY AND THE MIRE OF HUMAN VEINS.

BEFORE ME FLOATS AN IMAGE, MAN OR SHADE,
SHADE MORE THAN MAN, MORE IMAGE THAN A SHADE;
FOR HADES' BOBBIN BOUND IN MUMMY-CLOTH
MAY UNWIND THE WINDING PATH;
A MOUTH THAT HAS NO MOISTURE AND NO BREATH
BREATHELESS MOUTHS MAY SUMMON;
I HAIL THE SUPERHUMAN;
I CALL IT DEATH-IN-LIFE AND LIFE-IN-DEATH.

MIRACLE, BIRD OR GOLDEN HANDIWORK,
MORE MIRACLE THAN BIRD OR HANDIWORK,
PLANTED ON THE STAR-LIT GOLDEN BOUGH,
CAN LIKE THE COCKS OF HADES CROW,
OR, BY THE MOON EMBITTERED, SCORN ALOUD
IN GLORY OF CHANGELESS METAL
COMMON BIRD OR PETAL
AND ALL COMPLEXITIES OF MIRE OR BLOOD.

AT MIDNIGHT ON THE EMPEROR'S PAVEMENT FLIT
FLAMES THAT NO FAGGOT FEEDS, NOR STEEL HAS LIT,
NOR STORM DISTURBS, FLAMES BEGOTTEN OF FLAME,
WHERE BLOOD-BEGOTTEN SPIRITS COME
AND ALL COMPLEXITIES OF FURY LEAVE,
DYING INTO A DANCE,
AN AGONY OF TRANCE,
AN AGONY OF FLAME THAT CANNOT SINGE A SLEEVE.

ASTRADDLE ON THE DOLPHIN'S MIRE AND BLOOD,
SPIRIT AFTER SPIRIT! THE SMITHIES BREAK THE FLOOD,
THE GOLDEN SMITHIES OF THE EMPEROR!
MARBLES OF THE DANCING FLOOR
BREAK BITTER PURIES OF COMPLEXITY,
THOSE IMAGES THAT YET
FRESH IMAGES BEGET,
THAT DOLPHIN-TORN, THAT GONG-TORmented SEA.

SAILING TO BYZANTIUM
AN AGED MAN IS BUT A PALTRY THING,
A TATTERED COAT UPON A STICK, UNLESS
SOUL CLAP ITS HANDS AND SING, AND LOUDER SING
FOR EVERY TATTER IN ITS MORTAL DRESS,
NOR IS TERE SINGING SCHOOL BUT STUDYING
MONUMENTS OF ITS OWN MAGNIFICENCE;
AND THEREFORE I HAVE SAILED THE SEAS AND COME
TO THE HOLY CITY OF BYZANTIUM.

THE MASK
'PUT OFF THAT MASK OF BURNING GOLD
WITH EMERALD EYES.'
'O NO, MY DEAR, YOU MAKE SO BOLD
TO FIND IF HEARTS BE WILD AND WISE,
AND YET NOT COLD.'

COMMAND:

RETRIEVE THE TITLES OF ALL POEMS WITH THE FOLLOWING CHARACTERISTICS:
- BEAUTY | MEN;

** NON EXISTANT KEYWORD IN BOOLEAN EXPRESSION.
COMMAND COULD NOT BE PROCESSED

COMMAND:

RETRIEVE THE BEAUTIFUL AND STIRRING TEXT OF THE FIRST 3 POEMS WITH
THE FOLLOWING CHARACTERISTICS:
{{(BLOOD|DANCE) | DEATH} | (HEART | WOOD)};

THE TITLES AND TEXTS ARE:

THE SONG OF WANDERING AENGUS

I WENT OUT TO THE HAZEL WOOD,
BECAUSE A FIRE WAS IN MY HEAD,
AND CUT AND PEELED A HAZEL WAND,
AND HOOKED A BERRY TO A THREAD;
AND WHEN WHITE MOTHS WERE ON THE WING,
AND MOTHT-LIKE STARS WERE FLICKERING OUT,
I DROPPED THE BERRY IN A STREAM
AND CAUGHT A LITTLE SILVER TROUT.

WHEN I HAD LAID IT ON THE FLOOR
I WENT TO BLOW THE FIRE AFAME,
BUT SOMETHING RUSTLED ON THE FLOOR,
AND SOME ONE CALLED ME BY MY NAME:
IT HAD BECOME A GLIMMERING GIRL
WITH APPLE BLOSSOM IN HER HAIR
WHO CALLED ME BY MY NAME AND RAN
AND FADED THROUGH THE BRIGHTENING AIR.

THOUGH I AM OLD WITH WANDERING
THROUGH HOLLOW LANDS AND HILLY LANDS,
I WILL FIND OUT WHERE SHE HAS GONE,
AND KISS HER LIPS AND TAKE HER HANDS;
AND WALK AMONG LONG DAPPLED GRASS,
AND PLUCK TILL TIME AND TIMES ARE DONE
THE SILVER APPLES OF THE MOON,
THE GOLDEN APPLES OF THE SUN.

BYZANTIUM

THE UNPURGED IMAGES OF DAY RECEDE;
THE EMPEROR'S DRUNKEN SOLDIERY ARE ABED;
NIGHT RESONANCE RECEDES, NIGHT-WALKERS' SONG
AFTER GREAT CATHEDRAL GONG;
A STARLIT OR A MOONLIT DOME DISDAINS
ALL THAT MAN IS,
ALL MERE COMPLEXITIES,
THE FURY AND THE MIRE OF HUMAN VEINS.

BEFORE ME FLOATS AN IMAGE, MAN OR SHADE,
SHADE MORE THAN MAN, MORE IMAGE THAN A SHADE;
FOR HADES' BOBBIN BOUND IN MUMMY-CLOTH
MAY UNWIND THE WINDING PATH;
A MOUTH THAT HAS NO MOISTURE AND NO BREATHE
BREATHLESS MOUTHS MAY SUMMON;
I HAIL THE SUPERHUMAN;
I CALL IT DEATH-IN-LIFE AND LIFE-IN-DEATH.

MIRACLE, BIRD OR GOLDEN HANDIWORK,
MORE MIRACLE THAN BIRD OR HANDIWORK,
PLANTED ON THE STAR-LIT GOLDEN BOUGH,
CAN LIKE THE COCKS OF HADES CROW,
OR, BY THE MOON EMBITTERED, SCORN ALOUD
IN GLORY OF CHANGELESS METAL
COMMON BIRD OR PETAL
AND ALL COMPLEXITIES OF MIRE OR BLOOD.

AT MIDNIGHT ON THE EMPEROR'S PAVEMENT FLIT
FLAMES THAT NO FAGGOT FEEDS, NOR STEEL HAS LIT,
NOR STORM DISTURBS, FLAMES BEGOTTEN OF FLAME,
WHERE BLOOD-BEGOTTEN SPIRITS COME
AND ALL COMPLEXITIES OF FURY LEAVE,
DYING INTO A DANCE,
AN AGONY OF TRANCE,
AN AGONY OF FLAME THAT CANNOT SINGE A SLEEVE.

ASTRADDLE ON THE DOLPHIN'S MIRE AND BLOOD,
SPIRIT AFTER SPIRIT! THE SMITHIES BREAK THE FLOOD,
THE GOLDEN SMITHIES OF THE EMPEROR!
MARBLES OF THE DANCING FLOOR
BREAK BITTER FURIES OF COMPLEXITY,
THOSE IMAGES THAT YET
FRESH IMAGES BEGET,
THAT DOLPHIN-TORN, THAT GONG-TORMENTED SEA.

THE CIRCUS ANIMALS' DESERTION
THOSE MASTERFUL IMAGES BECAUSE COMPLETE
Grew in pure mind, but out of what began?
A mound of refuse or the sweepings of a street,
old kettles, old bottles, and a broken can,
old iron, old bones, old rags, that raving slut
who keeps the till. Now that my ladders start,
in the foul rag-and-bone shop of the heart.

Command:

initialize : beauty, blood , dance, death, heart, moon, gyre,
wood, sea;

** the keyword directory has been re-initialized.

all of the preceding keywords have been added.

Command:

delete the first 5 poems with the following characteristics:
((blood | dance) | death ) | (heart | wood));

the poem(s) with the following title(s) have been deleted:

the song of wandering aengus
byzantium
the circus animals' desertion
the death of the hare
under ben bulben

Command:

add keyword: murmur, cold;

all of the preceding keywords have been added.

Command:

retrieve for me the first 2 poems with the following characteristics:
blood | death &→ cold;

the titles and texts are:

the second coming

turning and turning in the widening gyre
the falcon cannot hear the falconer;
things fall apart; the centre cannot hold;
mere anarchy is loosed upon the world,
the blood-dimmed tide is loosed, and everywhere
the ceremony of innocence is drowned;
the best lack all conviction, while the worst
ARE FULL OF PASSIONATE INTENSITY.

THE GYRES

THE GYRES! THE GYRES! OLD ROCKY FACE, LOCK FORTH;
THINGS THOUGHT TOO LONG CAN BE NO LONGER THOUGHT,
FOR BEAUTY DIES OF BEAUTY, WORTH OF WORTH,
AND ANCIENT LINEAMENTS ARE BLOTTED OUT.
IRRATIONAL STREAMS OF BLOOD ARE STAINING EARTH;
EMPEDOCLES HAS THROWN ALL THINGS ABOUT;
HECTOR IS DEAD AND THERE'S A LIGHT IN TROY;
WE THAT LOOK ON BUT LAUGH IN TRAGIC JOY.

COMMAND:

RETRIEVE THE POEMS WITH THE CHARACTERISTIC: DORK;

** NON EXISTANT KEYWORD IN BOOLEAN EXPRESSION.
COMMAND COULD NOT BE PROCESSED

COMMAND:

RETRIEVE THE POEMS WITH THE FOLLOWING CHARACTERISTICS: ( DANCE.
)))));

** SYNTAX ERROR IN BOOLEAN EXPRESSION.
COMMAND COULD NOT BE PROCESSED

COMMAND:

DISPLAY PSW;

** NO COMMAND KEYWORDS WERE FOUND BEFORE THE FIRST SEMICOLON.
THE COMMAND STRING HAS BEEN IGNORED.

COMMAND:

DORK FORK MUNG DUNG
DETERMINE KEYWORD: INTERDATA RETRIEVE 9 POEMS WITH THE FOLLOWING KEYWORDS:
BLOOD & 1;

THE ABOVE KEYWORDS HAVE BEEN DELETED WITH THE EXCEPTION OF:

INTERDATA
RETRIEVE
9
POEMS
WITH
THE
FOLLOWING
KEYWORDS

** KEYWORD NOT FOUND
** KEYWORD NOT FOUND
** KEYWORD NOT FOUND
** KEYWORD NOT FOUND
** KEYWORD NOT FOUND
** KEYWORD NOT FOUND
** KEYWORD NOT FOUND
** KEYWORD NOT FOUND
** KEYWORD NOT FOUND
** KEYWORD NOT FOUND
COMMAND:

DELETE 3 POEMS WITH THE FOLLOWING CHARACTERISTICS OLD & MOON;

** THE COLON IS MISSING BETWEEN THE COMMAND AND ITS OPERANDS. THE COMMAND STRING HAS BEEN IGNORED.  

COMMAND:

ADD KEYWORDS: MOON, I;  

** THE PRECEDING KEYWORDS HAVE BEEN ADDED WITH THE EXCEPTION OF THOSE LISTED BELOW  
** DUPLICATE KEYWORD MOON  

COMMAND:

RETRIEVE THE POEMS WITH THE FOLLOWING CHARACTERISTICS:
MOON;

THE TITLE AND TEXT IS:

THE PHASES OF THE MOON

AN OLD MAN COCKED HIS EAR UPON A BRIDGE;  
HE AND HIS FRIEND, THEIR FACES TO THE SOUTH,  
HAD TROD THE UNEVEN ROAD. THEIR BOOTS WERE SOILED,  
THEIR CONNEMARA CLOTH WORN OUT OF SHAPE;  
THEY HAD KEPT A STEADY PACE AS THOUGH THEIR BEDS,  
DESPITE A DWINDLING AND LATE-RISEN MOON,  
WERE DISTANT STILL. AN OLD MAN COCKED HIS EAR.

COMMAND:

DELETE THE POEMS WITH THE FOLLOWING CHARACTERISTICS: MOON;

THE POEM(S) WITH THE FOLLOWING TITLE(S) HAVE BEEN DELETED:

THE PHASES OF THE MOON

COMMAND:

RETRIEVE

THE POEMS WITH THE FOLLOWING TITLE: THE GYRES;

THE TITLE AND TEXT IS:
THE GYRES

THE GYRES! THE GYRES! OLD ROCKY FACE, LOOK FORTH;
THINGS THOUGHT TOO LONG CAN BE NO LONGER THOUGHT,
FOR BEAUTY DIES OF BEAUTY, WORTH OF WORTH,
AND ANCIENT LINEAMENTS ARE BLOTTED OUT.
IRRATIONAL STREAMS OF BLOOD ARE STAINING EARTH;
EMPEOCLES HAS THROWN ALL THINGS ABOUT;
HECTOR IS DEAD AND THERE'S A LIGHT IN TRICY;
WE THAT LOOK ON BUT LAUGH IN TRAGIC JOY.

COMMAND:

DELETE KEYWORD: KISS, MOON, KISS;

THE ABOVE KEYWORDS HAVE BEEN DELETED WITH THE EXCEPTION OF:

KISS
KISS

** KEYWORD NOT FOUND
** KEYWORD NOT FOUND

COMMAND:

RETRIEVE ALL POEMS WITH THE FOLLOWING CHARACTERISTICS: MOON;

** NON EXISTANT KEYWORD IN BOOLEAN EXPRESSION.
COMMAND COULD NOT BE PROCESSED

COMMAND:

DELETE KEYWORD: COLD, GIRL;

THE ABOVE KEYWORDS HAVE BEEN DELETED WITH THE EXCEPTION OF:

GIRL

** KEYWORD NOT FOUND

COMMAND:

RETRIEVE THE NUMBER OF THE POEMS WITH THE FOLLOWING CHARACTERISTICS:
DEATH | SEA;

ONE POEM WAS FOUND WITH THE SPECIFIED TITLE OR CHARACTERISTICS.

COMMAND:

RETRIEVE THE NUMBER OF THE POEMS WITH THE FOLLOWING CHARACTERISTICS:
DEATH SEA |;

** SYNTAX ERROR IN BOOLEAN EXPRESSION.
COMMAND COULD NOT BE PROCESSED

COMMAND:
ADD POEM:
HE WISHES HIS BELOVED WERE DEAD
WERE YOU BUY LYING COLD AND DEAD,
AND LIGHTS WERE PALING OUT OF THE WEST,
YOU WOULD COME HITHER, AND BEND YOUR HEAD,
AND I WOULD LAY MY HEAD ON YOUR BREAST,
AND YOU WOULD MURMUR TENDER WORDS,
FORGIVING ME BECAUSE YOU WERE DEAD;
NOW WOULD YOU RISE AND HASTEN AWAY,
THOUGH YOU HAVE THE WILL OF THE WILD BIRDS,
BUT KNOW YOUR HAIR WAS BOUND AND WOUND
ABOUT THE STARS AND MOON AND SUN.
OH WOULD, BELOVED, THAT YOU LAY
UNDER THE DARK LEAVES IN THE GROUND
WHILE LIGHTS WERE PALING ONE BY ONE.

THE POEM WITH THE TITLE: HE WISHES HIS BELOVED WERE DEAD
HAS BEEN ADDED.

COMMAND:

RETRIEVE ALL POEMS WITH THE FOLLOWING CHARACTERISTICS:
DEATH

ISEA:

** SYNTAX ERROR IN BOOLEAN EXPRESSION.
COMMAND COULD NOT BE PROCESSED

COMMAND:

RETRIEVE THE TEXT OF ALL THE POEMS WITH THE FOLLOWING CHARACTERISTICS:
MURMUR;

THE TITLE AND TEXT IS:

HE WISHES HIS BELOVED WERE DEAD

WERE YOU BUY LYING COLD AND DEAD,
AND LIGHTS WERE PALING OUT OF THE WEST,
YOU WOULD COME HITHER, AND BEND YOUR HEAD,
AND I WOULD LAY MY HEAD ON YOUR BREAST,
AND YOU WOULD MURMUR TENDER WORDS,
FORGIVING ME BECAUSE YOU WERE DEAD;
NOW WOULD YOU RISE AND HASTEN AWAY,
THOUGH YOU HAVE THE WILL OF THE WILD BIRDS,
BUT KNOW YOUR HAIR WAS BOUND AND WOUND
ABOUT THE STARS AND MOON AND SUN.
OH WOULD, BELOVED, THAT YOU LAY
UNDER THE DARK LEAVES IN THE GROUND
WHILE LIGHTS WERE PALING ONE BY ONE.
THE IR USER'S GUIDE

A Guide to the Information Retrieval System Version 1.0

by R.F. Gurwitz and R.T. Fleming
GENERAL DESCRIPTION

The Information Retrieval System (IR) is a program which can be used to set up and organize a data base. It employs a simple, free form, command language, by which the user can add, delete, or retrieve items to and from the data base. The system also provides a facility for indexing the data by a group of keywords. These keywords are associated with each member of the data base, and they can be retrieved or deleted by specifying a logical (boolean) expression consisting of the keywords and the logical operators and (\&) or (\|) and not (\¬). Thus, one would be able to retrieve or delete all members of the data base which were associated with the keywords (a) and not (b), etc. A facility exists for adding a deleting these keywords.

The data managed by IR consists of groups of lines of text called members. Each member consists of a title and a variable number of lines of text. The members are organized via a data base which contains locating information that enables the retrieval of the text of each member from the file in which it resides. As each member is added or as each new keyword is added, the members are scanned for the presence of the keywords. These and information associating the keywords with the members are also maintained in the data base.

The current implementation of IR considers the members poems, however this need not be the case; members could be legal briefs, scientific texts, etc. Also, this implementation is made to run on an IBM System 360 or 370 machine, under OS. However, with certain changes, it could be made to run as an interactive system under an operating system like CP/CMS or VM/370. Thus, processing sessions illustrated in this user's guide are non-interactive batch mode type. For further information as to the structure and internal operation of the system, the user should consult the IR PROGRAMMER'S GUIDE and the IR MODULAR SPECIFICATIONS. For the specification of the IR system and command language, see the specification sheet for the INFORMATION RETRIEVAL PROJECT.

What follows is a description of the use of the Information Retrieval System and its command language.
1 ORGANIZATION OF DATA

The basic unit of data that IR deals with is the member. Each member consists of the following. The first line of a member is considered the title of that member. Since the members are organized by title, duplicate titles are not allowed. The title must not be longer than 80 characters, and must occupy one card image. Continuation cards are not allowed. Any imbedded blanks in the title are compressed to one blank when the title is processed. Thus, the titles:

THE LAZY DOG JUMPED OVER, THE QUICK BROWN FOX.

and

THE LAZY DOG JUMPED OVER, THE QUICK BROWN FOX.

are considered identically by the system.

The rest of the member consists of any number of 80 character lines, one line per card image. This is called the text of the member. The text is not modified or reformatted in any way when it is added to the data set.

The text of the data is kept on an CS disk in a file. (See FILES). This file contains only the text of the members and not their titles. The titles are contained in the data base itself.

1.1 THE DATA BASE

The text is organized, indexed and managed by means of a data base consisting of locator data for each member and for the keywords in the system. (For the exact organization of the data base see the PROGRAMMER'S GUIDE). The data base is maintained in a file separate from the file containing the text (See FILES). Each member is referred to by title within the directory of members. The title of the poem references a table which locates an individual data base element for each member in the system. Each member is also sequentially organized as it is added to the system. The data base element (member element), contains the title of the member, and indexing information in the text file for text retrieval. In addition, each member is linked to the other members in the data base as described above, and the necessary locator data for that is also contained in the element. The sequential linkage of elements facilitates keyword searches. (Note that if a new keyword is added to the system, the entire text file must be scanned. If a member is added to the system, it must be scanned for all keywords currently in the system.)

The keywords in the system also have individual elements in the system. They are organized by keyword name, as well as sequentially in a manner similar to the members. Each keyword is maintained by the insertion of a key header into the data base, which contains the keyword name, and locator information for the linking as described. It also contains a list of member references. These reference elements (keylists), consist of indexing information which cross-references the keywords with the member elements, so
that each keyword has a list of members associated with it.

Thus, the data base is really two directories of cross referencing information. One directory, references members with their associated texts on file. The other directory, references keywords to the members associated with them. In this directory, we are referencing the member directory, whereas in the member directory, we reference the actual texts.
2 FILES

The data base and text are kept in files that must be accessed with each use of the system. Also needed for processing are a member input file, used for initializing a data base, and a command file for inputting commands for each processing session. A print file is also used, for printing the output generated in each session. These needed files are described briefly in this section. For further details consult the PROGRAMMER'S GUIDE and the section in this guide on JCL.

The data base is stored between sessions in a file called DBASE. The file is read from disk at the beginning of a non-initial session and is written back at the end of the session. Thus, changes in the data base made in the last session will be reflected in the contents of this file at the beginning of the next session. When a new data base is set up, this file is overwritten with the new data base. If at any time the data base fills to capacity (for current implementation specifications and size restrictions, see the PROGRAMMER'S GUIDE), the data base is immediately written back to the disk for later recovery.

The text is kept on a separate file called TEXT. This file is modified when members are added to it. The file is not modified if members are deleted from it, only the data base information is changed in the current implementation. Thus, the text of deleted members is retained in the system but cannot be accessed.

An input file is used for system initialization. This file is known as POEM and contains the titles and texts of the members to be added to the system at the beginning of the data base setup. For a description of the format of this file, see the section on USE OF THE SYSTEM (SETUP). For format of the command and print files used by IR, see the sections on JCL and on USE OF THE SYSTEM (COMMAND LANGUAGE).

In the current implementation, the files are maintained on an OS disk for storage between sessions. For organization and disk access see the section on JCL and refer also to the appropriate OS manuals.
3 USE OF THE SYSTEM

The system is used in two modes. To initiate processing, a data base must be set up with an initial set of members (if desired) and the appropriate data base directory information. After the initial setup, the system is used in processing sessions. These sessions consist of the processing of a command file containing commands to modify or retrieve members in the data base. This section contains a description of how the system is to be used to catalogue and access information.

3.1 SETTING UP A DATA BASE

In the initial use of the system, the data base must be initialized as described above. This procedure should also be done if the user wishes to erase his current data base and replace it with a new one. To invoke the initialization procedure, the user should specify the SETUP option in the JCL for the run (see JCL). When this is done, all previous information in the data base is erased, and the files are overwritten. A set of initial members may be added to the newly created data base. These members should be contained in file POEM and they should be formatted as follows:

```
**************....**********  (A card of 80 asterisks)
< TITLE OF MEMBER 1 >
< TEXT OF MEMBER 1 >
```

```
< TEXT OF MEMBER 1 >
```

```
**************....**********  (80 asterisks)
< TITLE OF MEMBER 2 >
< TEXT OF MEMBER 2 >
```

and so on for as many members as the user wishes to add. These members will be added, but the text will not be scanned for keywords, since there are none in the system to start with. The rest of the initialization procedure consists of setting up the directories for later entries and reinitializing various other parameters. A setup run may be made without adding members, but the input file (POEM) must be available to the system regardless. When a setup run is made, the system will flag it by indicating

```
**SETUP OPTION IN EFFECT**
```

on the header page. The system will respond to the addition of members in the initialization procedure by printing one of the following three messages followed by the text added.

```
THE FOLLOWING POEM HAS BEEN ADDED TO THE TEXT FILE:
** THE FOLLOWING POEM DUPLICATEES TEXT ALREADY ON FILE AND WAS NOT ADDEED:
** INVALID TITLE: THE FOLLOWING POEM IS DISREGARDED.
```

The initialization is then complete, and the session
continues with command processing as in a normal session.

3.2 THE PROCESSING SESSION

Any modification or retrieval of information is done in the processing session. A session consists of the processing of the command file input to IR. This file (SYSIN) contains commands formatted according to the rules of the command language (see COMMAND LANGUAGE). At the beginning of the session, the data base file (DBASE) is read from disk and parameters are initialized. The parameters HIGH_TEXT and LOW_NUMBER which affect the output format of specified text retrievals (see COMMAND LANGUAGE) are defaulted to 5 and 20, respectively. The data and data base should look exactly as they did at the end of the previous session.

The command file is read, and each is processed in turn. Output from the program consists of a header page, giving the time, date, and the number of the run with the current data base. On the pages following, the commands are printed as read, followed by the system response which includes any system error messages. When the end of the command file is reached, a completion message is printed, the data base is saved on disk, and the execution of the program terminates. Any changes made during the session should be reflected in the status of the data base and text files.
4 THE COMMAND LANGUAGE

The IR command language is a free format, keyword oriented language which enables manipulation of the data base. There are six commands: INITIALIZE, LIST, SET, ADD, DELETE, and RETRIEVE. This section describes the use and syntax of each command, general syntax of the language, as well as error messages generated in the command processor are also described.

4.1 GENERAL SYNTAX CONSIDERATIONS

The command processor accepts free format commands and works on the so-called principle of the finite state automata. That is, it scans for a recognizable keyword (command word), ignoring non-keywords, and interprets the keyword, moving into a different state, which looks for the next expected keyword. Thus, a syntax is maintained, which, if violated, will result in a syntax error. The following rules are generally observed in commands.

1. Commands can contain imbedded blanks and any number of non-keywords.

2. Command keywords may not contain imbedded blanks.

3. Commands may extend across line boundaries, but keywords may not be split across boundaries.

4. All commands except for the ADD POEM command must end with a semi-colon (;).

If any of these rules are broken, they will result in an appropriate error message and the command will be disregarded. Following, is an explanation of command syntax error messages.

4.1.1 COMMAND SYNTAX ERRORS

** END OF FILE ENCOUNTERED BEFORE COMMAND STRING TERMINATION CHARACTER.

Explanation: Command file ends before a semi-colon or delimiting asterisks can be found. Probable user error in forgetting to code a termination character. One is assumed.

** NO COMMAND KEYWORDS WERE FOUND BEFORE THE FIRST SEMI-COLON. ** NO COMMAND WORDS WERE FOUND BEFORE THE FIRST COLON.

Explanation: No command was found.

** THE COMMAND IS MISSING KEYWORDS BEFORE THE FIRST COLON.
** THE COMMAND IS MISSING KEYWORDS BEFORE THE FIRST SEMI-COLON.
**THE COLON IS MISSING BETWEEN A COMMAND AND ITS OPERANDS.**

Explanation: A character other than a colon was found after a command keyword that uses operands. The command is ignored.

**IN THE ADD POEM COMMAND, THE END OF POEM DELIMITER WAS ENCOUNTERED BEFORE THE POEM.**

Explanation: No title or text was found after the add poem command.

4.2 COMMAND USAGE AND SYNTAX

The following conventions are observed in this section. Blanks may be inserted anywhere in a command line. Where a comma is used to separate elements of an operand list, a blank may be substituted. Angle brackets (<>') surround examples of keywords, expressions, or parameters. These serve to delimit in the examples and need not be typed in the actual command. Defaults are underlined. Substitute keyword options are shown separated with a vertical line (|).

4.2.1 INITIALIZE

INITIALIZE :<keyword 1> , <keyword 2> , ... , <keyword n> ;

Usage: The initialize command erases the current contents of the keyword directory and substitutes those keywords that are specified. Any members in the system are rescanned for the presence of the keywords and directory entries are created for them.

Response: **THE KEYWORD DIRECTORY HAS BEEN RE-INITIALIZED.**

Errors:

**DUPLICATE KEYWORD SPECIFIED.**

**KEYWORD TOO LONG.**

**SPECIAL CHARACTER.**
4.2.2 LIST

<table>
<thead>
<tr>
<th>TITLES</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST</td>
</tr>
<tr>
<td>KEYWORDS</td>
</tr>
<tr>
<td>PARAMETERS</td>
</tr>
<tr>
<td>ALL</td>
</tr>
</tbody>
</table>

Usage: A listing of the current members, keywords and associated members, system parameters, HIGH_TEXT and LOW_NUMBER or all three.
4.2.3 SET

| HIGH_TEXT | : <value> ; |
| LOW_NUMBER |

Usage: This command enables the user to reset the values for the parameters HIGH_TEXT and LOW_NUMBER. These control the output of members for the RETRIEVE command according to the following rules:

1. If the set of members contains less than HIGH_TEXT elements to be printed, the text is printed.

2. If the set contains between HIGH_TEXT and LOW_NUMBER, inclusive, the titles are printed.

3. If the set contains more than LOW_NUMBER members, only the number of members satisfying the request is printed.

These rules are only observed if the output option of the RETRIEVE command is unspecified. The parameters default to 5 and 20 at the beginning of each processing session.

Value, must be a valid number between 1 and 30. The following restrictions are observed: HIGH_TEXT < LOW_NUMBER always. If the command does not observe this restriction, or if invalid values are given, the command will be ignored.

System response:
HIGH_TEXT RESET TO <value>.
LOW_NUMBER RESET TO <value>.

Errors:
** NO DATA SPECIFIED. COMMAND IGNORED.
** NON-NUMERIC DATA SPECIFIED. COMMAND IGNORED.
** SPECIFIED DATA IS OUT OF BOUNDS. COMMAND IGNORED.
** SPECIFIED HIGH_TEXT NOT LESS THAN LOW_NUMBER. COMMAND IGNORED.
** SPECIFIED LOW_NUMBER NOT GREATER THAN HIGH TEXT. COMMAND IGNORED.
4.2.4 ADD

```
| KEYWORD[S] | ADD | : <data> ; |
|  |    | POEM |
```

Usage: This command allows the addition of keywords or members to the system. If keywords are added, data is `<keyword 1>`, `<keyword 2>`, `<keyword n>`. The current members are scanned for the presence of the new keywords and new directory entries are made for them. If POEM is specified, data is the following:

```
<TITLE OF MEMBER>
<TEXT OF MEMBER>
*************.....*************** (CARD OF 80 ASTERISKs).
```

In this case, the member is added to the directory, its text written to the text file, and the text scanned for the presence of current keywords, with the keyword directory updated as necessary. No semi-colon follows the command in this format.

Response:
THE POEM WITH TITLE: <title> HAS BEEN ADDED.
THE ABOVE KEYWORDS HAVE BEEN ADDED [WITH THE EXCEPTION OF THOSE LISTED BELOW: ]

Errors:

** DUPLICATE TITLE GIVEN.
** NO TITLE GIVEN.
** POEM NOT ADDED DUE TO AREA OVERFLOW.
** DUPLICATE KEYWORD.
** KEYWORD TOO LONG.
** SPECIAL CHARACTER.
** NO DATA GIVEN.
### 4.2.5 DELETE

<table>
<thead>
<tr>
<th>KEYWORD[S]</th>
<th>TITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;no.&gt;</td>
<td></td>
</tr>
<tr>
<td>POEM[S]</td>
<td></td>
</tr>
<tr>
<td>ALL</td>
<td>CHARACTERISTICS</td>
</tr>
</tbody>
</table>

**Usage:** The delete command allows deletion of keywords or poems by the system. If keywords are specified, the keyword entries in the directory will be erased. The data looks like: `<keyword 1>, <keyword 2>, <keyword n>`. If poems are specified, the data may be either a poem title or a boolean expression, depending on the specification of TITLE or CHARACTERISTICS. If TITLE is specified, the `<n>` members (1) with the specified title will be deleted from the directory. Otherwise, if characteristics are specified, the boolean expression will be evaluated and `<n>` members conforming to the given characteristics will be deleted.

**Response:**

NO POEMS WERE FOUND WITH THE SPECIFIED TITLE OR CHARACTERISTICS.
THE ABOVE KEYWORDS HAVE BEEN DELETED [WITH THE EXCEPTION OF THOSE LISTED BELOW]
THE POEM(S) WITH THE FOLLOWING TITLE(S) HAVE BEEN DELETED:
<TITLE>

**Errors:**

** INVALID POEM TITLE.**
** NON-EXISTANT POEM SPECIFIED.**
** KEYWORD TOO LONG.**
** KEYWORD NOT FOUND.**

Any errors in the boolean expression will be flagged. (See BOOLEAN SYNTAX)
4.2.6 RETRIEVE

| TITLE[S]  | [ | <no.> | ] POEM[S] | TITLE  | : <expression>
| NUMBER   |   | ALL   | CHARACTERISTICS |

Usage: The retrieve command will retrieve the title, text, or number of members specified either by title or characteristics. If title is specified, the <n> members (1), with the specified title will be printed. Otherwise, those <n> members which satisfy the boolean expression will be printed. The output format may or may not be specified. If specified, the title, text or number is outputted. Otherwise, the format is determined by the parameters HIGH_TEXT and LOW_NUMBER as outlined in the section on the SET command.

Responses:

The system will respond with the requested information, if available. The only errors are those which are reflected from the boolean expression. (See BOOLEAN EXPRESSION SYNTAX).
4.3 BOOLEAN EXPRESSION SYNTAX

When the characteristics option has been specified in the DELETE and RETRIEVE commands, the result of a specified boolean expression is taken as the specification. Expressions consist of groups of keywords and logical operators forming a logical clause, which may be of an unspecified level of complexities. The only valid operators are the and operator, $\&$, the or operator, $\mid$, the not operator, $\neg$, and parenthesis. The and and or operators are dyadic, that is they appear legally as $<\text{keyword}> <\text{operator}> <\text{keyword}>$. The not operator is monadic. That is, $<\text{operator}> <\text{keyword}>$. Expressions may contain imbedded blanks. The expressions must have balanced parentheses. The precedence of operators is $\neg, \&, \mid$, from lowest to highest. Precedence is overridden by the use of parenthesis. Invalid keywords, either too long or non-existent are errors. Any error in the expression will cause the command to be ignored.

Errors:

** SYNTAX ERROR IN BOOLEAN EXPRESSION.
** NON EXISTANT KEYWORD IN BOOLEAN EXPRESSION.
The JCL to run the system is the following:

//RUN JOB (acct. no.,1,5),name,MSGLEVEL=1,CLASS=K // EXEC PL1PFG,PARM,GO='EP=IENTRY,LET/user parm' //LOADIN DD DSN=ir disk member,VOL=SER=diskname, // UNIT=2314,DISP=(OLD,KEEP) //POEMS DD DSN=disk member,VOL=SER=disk name,DISP=SHR, // UNIT=2314 //DBASE DD DSN=database member name,VOL=SER=disk name, // UNIT=2314,DISP=(OLD,KEEP) //

Where user parm is SETUP for the setup run, and the appropriate disk and member names should be substituted.
INFORMATION RETRIEVAL SYSTEM
PROGRAMMER'S GUIDE

By R.T. Fleming and B.F. Gurwitz

IR Version 1.1 -- 1/12/76
IR FILES AND I/O

The IR system uses five files for input and output. Their usage is listed below and the names given are those used in the program.

DBASE stores the system database (i.e., poem and keyword directories) between runs. The file consists of one massive (28K bytes) U-format record, which gets read into the structure DIRECTS at the start of a run, and gets written out at the end.

POEM is the sequential file containing the poems to be used to initialize the system when the SETUP option is in effect. The logical record length is 80 bytes, and poems consist of a record with the title, followed by zero or more lines (records) of text. Delimiter cards of 80 asterisks separate poems. They may be omitted at the start and end of the file, and any number of them may appear between adjacent poems.

SYSIN is the sequential file containing the IR command stream. The logical record length is 80 bytes. Refer to the IR User's Guide for information on individual commands and their usage.

SYSPRINT is the system print file. In the current implementation of IR, the LINESIZE is set to the PL/I default of 120 characters.

TEXT stores the text of the poems maintained by IR. Titles are not stored here; they are kept in the poem directory. The TEXT file is logically divided into regions of fifteen 80 character lines. When poem text is added, it is always appended immediately after the last line. This means that if the last block was partially filled and in the file, it is read in and added to before any new blocks are written. Thus the only partially filled block is at the end of the file.

All reading and writing to and from the TEXT file is done by the READLIN and WRITELIN routines. A line of text is always referenced by a file line number, which corresponds to a region \((1,2,\ldots)\) and a line \((1,2,\ldots,15)\) within the region where the text can be found. READLIN and WRITELIN automatically handle this calculation and do any needed I/O to access the referenced line.

It is important to note that an I/O operation is not necessarily performed each time one of these routines is called. The CUR_REGION field of the external structure IRTEXTBL keeps track of the current region in core, and if the requested region is that region (for either READLIN or WRITELIN), then a read from the file is unnecessary. In the same way, regions are written out only when another region must be accessed or at the end of a run.
THE POEM AND KEYWORD DIRECTORIES

IR uses the inverted file technique of information retrieval. Poems can be referenced by either title or keywords, so the system maintains a list of titles in the poem directory and a list of keywords and keylists in the keyword directory. Each directory has its own area and hash table for accessing the data in the area.

The hashing algorithm for either poem titles or keywords is the same: the item to be hashed is broken into groups of four bytes (32 bits) and they are exclusive-or'ed together, then the result is packed to 24 bits by removing the leading two bits of each byte, and then that number is divided by 211 and one is added to the remainder. The final result is a FIXED BINARY (31,0) integer from 1 to 211 which is used to index one of the two (poem or keyword) hash tables. These contain offsets to the associated buckets where the data may be found.

When a poem is added to the system, its title is hashed to obtain the bucket where the new directory entry (POEMLIST_ELEMENT) should go. The entry is always added to the end of the bucket, so that a linear search must be used when looking through the bucket for the title. The POEMLIST_ELEMENT includes the title, and starting line number and number of lines in the file. In addition, there are chaining offsets to link an entry to adjacent ones in the bucket. Thus we have:

![Diagram of hash table and buckets](image)

**Figure 1**
The keyword directory is similar in concept but there are two kinds of data structures, **KEY_HEADERS** and **KEYLIST_ELEMENTS**, allocated within the area. For each keyword there is a **KEY_HEADER** which can be accessed through a hash table like the poem titles. Several of these may be chained together in a bucket, and a linear search is used to find a given keyword within the bucket. Buckets are generally only a few entries long, however, and this is not a problem. The header contains the keyword, chaining offsets to adjacent headers in the bucket, and an offset to the first **KEYLIST_ELEMENT** of the associated keylist. The keylist is a linked list of offsets to **POEMLIST_ELEMENTS** (**PLE's**) corresponding to those poems which have the keyword in them. Hence the keyword directory looks like:

![Figure 2](image)

The **KEYLIST_ELEMENTS** in any given keylist are linked together in ascending order by the binary value of the PLE offsets. This is reasonable since the offsets are simply fullword binary numbers which uniquely identify entries in the poem directory. This means that they can be sorted, and also used to access the poem directory directly without resorting to methods such as poem numbers, etc.

The reason that they are sorted is to allow Boolean operations (AND, OR, and NOT) to be performed on them. The NOT of a keylist is the list of PLE offsets which are in the directory and do not appear in the keylist to be negated. A "master" list of offsets is needed to do this, and it is accomplished by linking the **POEMLIST_ELEMENTS** sequentially by offset. Thus a PLE is really part of two linked lists: The one which makes up the bucket, and the one which makes up the sequential list of PLE's.

All the **KEY_HEADERS** are also linked together in one master list. This list is useful when all of the headers must be accessed (such as adding or deleting poems) and we want to avoid checking all 211 buckets. Unlike the PLE master list, the **KEY_HEADER** list is not maintained in any particular order; rather, entries are always added to the end of the list.
IR LIST MANAGEMENT

Some effort has been made in this implementation of IR to cut down on the often substantial overhead incurred by PL/I list processing. This has been done by avoiding the use of ALLOCATE and FREE statements, and instead using code which manages IR's own "free lists". When a list element is freed in this manner it becomes part of the IR free list. When one must be allocated, it will be taken from the IR free list (if any are there) rather than using ALLOCATE.

There are six kinds of list elements in IR:

<table>
<thead>
<tr>
<th>item</th>
<th>allocated in</th>
<th>free list pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>POEMLIST_ELEMENT</td>
<td>POEM_AREA</td>
<td>PLE_GL_OFFSET</td>
</tr>
<tr>
<td>KEY_HEADER</td>
<td>KEY_AREA</td>
<td>KHD_GL_OFFSET</td>
</tr>
<tr>
<td>KEYLIST_ELEMENT</td>
<td>KEY_AREA</td>
<td>KLE_GL_OFFSET</td>
</tr>
<tr>
<td>C3LIST_ELEMENT</td>
<td>free core</td>
<td>C3GLPTR</td>
</tr>
<tr>
<td>WORKLIST_ELEMENT</td>
<td>free core</td>
<td>W3GLPTR</td>
</tr>
<tr>
<td>WORKLIST_ELEMENT*</td>
<td>free core</td>
<td>-</td>
</tr>
</tbody>
</table>

The last item (marked with *) makes up a special list allocated and freed within the ADIKEYS module. It does not make use of the list management techniques described below, but the other five do. Furthermore, the methods of "freeing" and "allocating" are the same for all five, and we can show how they work with one general model.

Consider first the situation where we want to free an element. The list linkage in this example is one-way, and we have a free list pointer which points to the top of the list of "freed" elements. P1 points to the element to be freed (C in figure 3), and P2 points to the element before it.

![Figure 3](image1)

![Figure 4](image2)
The following PL/I code will give us figure 4, by chaining the list around item C and putting C on the free list:

```
P2->NEXT_LE_PTR = P1->NEXT_LE_PTR;
P1->NEXT_LE_PTR = FREE_LIST_PTR;
FREE_LIST_PTR = P1;
```

Obviously, when the chaining is more complicated (such as with a POEMLIST_ELEMENT) more code is needed to detach it from the list. However, the last two statements (which replace a PL/I FREE) remain the same: freed list elements are pushed onto the stack pointed to by FREE_LIST_PTR. Also note that no exception is needed when the stack is empty; in this case FREE_LIST_PTR is null.

Allocation is equally simple. Figure 5 shows the situation when we want to add an element to the bottom of a list. LE_PTR is the basing pointer for the element, and here it points to the last element in the list. We can allocate as follows:

```
P1 = LE_PTR;
IF FREE_LIST_POINTER = NULL:
    THEN ALLOCATE LIST_ELEMENT;
ELSE DO;
    LE_PTR = FREE_LIST_PTR; /* LIST BY POPPING */
    FREE_LIST_PTR = NEXT_LE; /* TOP LIST_ELEMENT */
END;
P1->NEXT_LE = LE_PTR; /* CHAIN TO PREVIOUS */
```

The result is shown in figure 6. The reader may wonder if the areas and free core eventually fill up with these "garbage" lists. This does not happen: the total number of allocations reaches a maximum after a time and all subsequent list elements will be taken from the free lists without any reference to the PL/I-OS routines.

![Figure 5](image1)

![Figure 6](image2)
MODULAR SPECIFICATION

By R.F. Gurwitz and R.T. Fleming

NAME:
IR

FUNCTION:
IR (Information Retrieval), is a system that allows manipulation and access to a data base of poems. IR enables a poem file to be built and organized in such a way that it can be accessed and updated by means of a simple command language. Poems can be referenced by title or by keyword attributes that are associated with each poem, by means of a command. These attributes may be specified in boolean expressions which allow poems to be deleted or retrieved. Poems or keyword attributes may be added or deleted individually at any time, while other features allow the data base to be listed and system parameters to be changed.

CALLED BY:
System

PARAMETERS:
Setup parameter is passed from system to indicate IR initialization.

CALLING SEQUENCE:
Standard OS linkage.

NORMAL EXITS:
Execution is complete when the command input file (SYSIN) reaches EOF.

ERROR CONDITIONS:
IR should not bomb on any input. Processing errors are reflected from execution routines by error message or similar action.

FILES:
SYSIN-- Command input file.
SYSPRINT-- Print output file.
DBASE-- Data base file.
TEXT-- Poem text file.
POEM-- Poem input file.

See attached sheet describing files for further details.

STORAGE UTILIZATION:
See attached sheet on data structures.

DESIGN CONSIDERATIONS:
IR is highly structured and uses modularization to the
fullest extent. Certain routines have been written in S/360 Assembler to optimize their efficiency and execution time. **IR** is a user oriented program; user convenience has been taken into account wherever possible.
NAME:
IR MAINLINE

FUNCTION:
The mainline is the main execution loop of IR. It opens all necessary files and handles end of file and area overflow. It reads and writes the data base from and to disk at the beginning and end of each processing session. If the setup parameter was passed to IR, it calls the SETUP routine. It then begins processing commands, calling COMSET to read them and parse them; and by reading the table returned from COMSET, calls the proper command processing routine to handle the command. The loop ends when the last command has been processed.

 CALLED BY:
 System

 PARAMETERS:
The setup parameter is passed from system to indicate IR initialization.

 NORMAL EXITS:
Execution ends when last command has been processed.

 FILES:
The mainline opens and closes the following files: SYSIN, SYSPRINT, DBASE, and TEXT.

 OUTPUT:
The mainline prints headers and trailers.

 EXTERNAL ROUTINES CALLED:
 SETUP, COMSET, IRADD, IRRERET, IBSET, IRLST

 INTERNAL ROUTINES CALLED:
 TITLE_PAGE -- Prints the title page.

 EXTERNAL DATA REFERENCED:
The mainline references the command table set up by COMSET. It also reads and writes the data base to and from disk.
MODULAR SPECIFICATION

NAME:
SETUP

FUNCTION:
This routine initializes the disk file of poems. SETUP reads in a series of poems and creates directory entries for them. It prints the text of the poems added and writes them to disk. SETUP is only called for system initialization. Setup also initializes file I/O parameters and the data base directories.

CALLED BY:
IR

PARAMETERS:
none

NORMAL EXITS:
Execution ends on EOF on the POEM file.

ERROR CONDITIONS:
Setup checks for duplicate or invalid poem titles.

FILES:
Setup reads the POEM file and writes the TEXT file.

INPUT:
The poem titles and text contained in file POEM are input to the routine.

OUTPUT:
SETUP prints the title and text of the poems it writes to disk, as well as any error messages for errors incurred when adding a poem.

INTERNAL ROUTINES CALLED:
none

EXTERNAL ROUTINES CALLED:
ADTITLE, WRITELN

EXTERNAL DATA REFERENCED:
SETUP initializes the data base directories and file TEXT.

GLOBAL ASSUMPTIONS & SIDE EFFECTS:
Any previous text in file TEXT is overwritten on disk.
MODULAR SPECIFICATION

NAME:
COMSET

FUNCTION:
COMSET sets up command strings for later processing. It reads them from SYSIN into a linked list, prints them and scans them for command keywords. The latter are set up into the global command keyword table COMTREE for use by the command processing routines.

CALLED BY:
IR

PARAMETERS:
one

CALLING SEQUENCE:
Invoked with CALL COMSET;

NORMAL EXITS:
Control returns from COMSET after a complete, valid command has been read, or end-of-file was reached on SYSIN.

FILES:
SYSIN and SYSPRINT are referenced by this routine.

INPUT:
See IR User's Guide for complete command descriptions and formats.

OUTPUT:
Command strings are printed in the form:

COMMAND STRING:

< line 1 >
< line 2 >
. .
< line n >

** < error message (optional) >

where the error message may result from a syntax error in the command string.

INTERNAL ROUTINES CALLED:
READ_CS does the initial read in of the command string.

READ_MORECS reads the rest of the command string, once READ_CS is finished with it.
EXTERNAL ROUTINES CALLED:
  TOKEN

EXTERNAL DATA REFERENCED:
  COMTREE is the 5 element array set up by COMSET with parameters for use by the command processing routines.
  END_COMMANDS is a flag within IRFLAGS set to indicate that no more command strings exist.
  IRCINFO is updated to reflect the current command string.

STORAGE UTILIZATION:
  Three major arrays are used in implementing the logic for command analysis. These are the list of keywords COMMAND_WORDS, the table of logical states NEXT_STATE_TABLE, and the branch table for the case statement which actually determines the logic flow.

ALGORITHM:
  During command analysis, the technique of finite state automata is used to process the command strings. The program passes through a sequence of logical 'states', advancing from one state to another according to the command keyword and the present state.
VERSION 5.4

OS/360 PL/I COMPILER (F)

PL/I F COMPILER OPTIONS SPECIFIED ARE AS FOLLOWS--

SM=(2,72,11),NOSTMT,NEST,ATR,XREF,NST,LOAD,NODECK

THE COMPLETE LIST OF OPTIONS USED DURING THIS COMPILATION IS--

- EBCDIC
- CHAR60
- NOMACRO
- SOURCE2
- NMCACOCK
- COMP
- SOURCE
- ATR
- XREF
- NOEXTRREF
- NOLIST
- LOAD
- NODECK
- FLAGW
- NOSTMT
- SIZE=0283892
- LNECNT=060
- OPT=01
- SRMTGIN=(002,072,001)
- NOEXTDIC
- NEST
- OPLIST
- SYNCHKT

*OPTIONS IN EFFECT*  EBCDIC,CHAR60,NOMACRO,SOURCE2,NMCACOCK,COMP,SOURCE,ATR,XREF,NOEXTRREF,NOLIST,LOAD,

*OPTIONS IN EFFECT*  NODECK,FLAGW,NOSTMT,SIZE=0283892,LNECNT=060,OPT=01,SRMTGIN=(002,072,001),NOEXTDIC,

*OPTIONS IN EFFECT*  NEST,OPLIST,SYNCHKT
/* IR -- INFORMATION RETRIEVAL SYSTEM */

/* IR: PROG(INSET) OPTIONS(MAIN); */

IR: PROG(INSET) OPTIONS(MAIN):

1

* THE IR MAINLINE DECLARES ALL EXTERNAL VARIABLES, AREAS,
  AND FILES. IT DEFINES ON UNITS FOR VARIOUS CONDITIONS THAT CAN
  ARISE IN EXECUTION, A PARAMETER CAN BE PASSED TO THE
  MAINLINE FROM THE OPERATING SYSTEM TO INDICATE IR SYSTEM
  INITIALIZATION. IF PASSED, THE SETUP ROUTINE IS CALLED TO READ
  IN THE INITIAL POEM FILE AND SET UP AREAS FOR PROCESSING.
  THE MAIN PROCESSING LOOP CONSISTS OF CALLING THE COMSET ROUTINE
  TO READ COMMANDS AND INTERPRET COMMAND KEYWORDS, AND
  DOING A CASE ON THE RESULT IN ORDER TO CALL THE COMMAND PROCESSING
  ROUTINES: IRAAD, IRDERET, IRLST, AND IRSET. EXECUTION
  ENDS WHEN THE LAST COMMAND IS EXECUTED OR ON AREA OVERFLOW.
  IN NORMAL USE (NOT INITIAL), THE MAINLINE READS THE DATABASE AREAS
  INTO CORE FROM DISK. IT ALWAYS WRITES THEM OUT
  TO DISK AT THE END OF EACH SESSION OR IF AREA OVERFLOW OCCURS. */

/* EXTERNAL ROUTINES */

2 1 DCL (SETUP,COMSET,IRAAD,IRDERET,IRSET,IRLST) ENTRY;

/* FILES */

3 1 DCL DBASE FILE RECORD SEQUENTIAL,
  /* DIRECTS* STORED HERE BETWEEN RUNS */
  TEXT FILE RECORD DIRECT ENVR(REGIONAL(1)),
  /* POEM TEXTS KEPT HERE */
  PCEM FILE RECORD SEQUENTIAL INPUT,
  /* THE SYSTEM IS INITIALLY SET UP USING THE
  POEMS IN HERE */
  SYSIN FILE RECORD SEQUENTIAL INPUT,
  /* COMMAND STRING FILE */
  SYSPRINT FILE STREAM OUTPUT PRINT;
  /* PRINTER */
STAT LEVEL NEST

/* THE FOLLOWING STRUCTURE CONTAINS THE POEM AND KEYWORD DIRECTORIES. */
IR 00340
IR READS THEM IN FROM THE DISK FILE DBASE AT THE BEGINNING OF EACH
RUN AND WRITES THEM OUT AT THE END. */
IR 00350
IR 00360

5 1
DCL 1 DIRECTS EXTERNAL,
IR 00370

2 POEM_DIRECTORY,
IR 00380

3 POEM_HASH_TABLE(211) OFFSET(POEM_AREA),
IR 00390
3 FIRST_SEQ_PLE OFFSET(POEM_AREA), /* PLE WITH LOWEST */
IR 00400
3 LAST_SEQ_PLE OFFSET(POEM_AREA), /* HIGHEST OFFSET */
IR 00410
3 PLE_GL_OFFSET OFFSET(POEM_AREA), /* PLE GARBAGE LIST */
IR 00420
3 POEM_AREA_BODY AREA((96000),
IR 00430

2 KEY_DIRECTORY,
IR 00440

3 KEY_HASH_TABLE(211) OFFSET(KEY_AREA),
IR 00450
3 FIRST_SEQ_KHD OFFSET(KEY_AREA),
IR 00460
3 LAST_SEQ_KHD OFFSET(KEY_AREA), /* HEADER GARBAGE LIST */
IR 00470
3 KHD_GL_OFFSET OFFSET(KEY_AREA), /* KEYLIST GARBAGE LIST */
IR 00480
3 KEY_AREA_BODY AREA((16800),
IR 00490

2 FILE_STATUS,
IR 00510

3 DIR_REGION FIXED BINARY(15,0), /* HIGHEST REGION */
IR 00520
3 DIR_LINE FIXED BINARY(15,0), /* LINE WHERE NEXT POEM */
IR 00530
3 RUN_NO FIXED DECIMAL(3,0), /* RUN NUMBER */
IR 00540

/* PA_PTR AND KA_PTR ARE SET TO POINT TO POEM_AREA_BODY AND
KEY_AREA_BODY RESPECTIVELY, SO THAT THE FOLLOWING BASED
AREAS MAY BE USED FOR THE POEM AND KEYWORD DIRECTORIES. */
IR 00560
IR 00570
IR 00580

6 1
DCL POEM_AREA AREA((96000)) BASED(PA_PTR),
IR 00590
KEY_AREA AREA((16800)) BASED(KA_PTR);
IR 00600

7 1
DCL PA_PTR POINTER EXTERNAL, /* BASES POEM_AREA */
IR 00610
KA_PTR POINTER EXTERNAL, /* BASES KEY_AREA */
IR 00620

/* THIS IS THE FREE LIST POINTER FOR THE WORKING KEYLIST */
IR 00630
*- STAT LEVEL NEST -*

/* THIS STRUCTURE CONTAINS THE SYSTEM PARAMETERS. */

```c
1 DCL 1 IRPAMS EXTERNAL.
2 HIGH_TEXT FIXED BINARY(15.0);   // IR 00650
2 LOW_NUMBER FIXED BINARY(15.0);   // IR 00670
2 LOOwn FIXED BINARY(15.0);        // IR 00680
```

/* THIS STRUCTURE CONTAINS THE SYSTEM FLAGS. */

```c
10 DCL 1 IRFLAGS EXTERNAL,
2 POE_EOF BIT(1),  // END-OF-FILE FOR POE_EOF */ IR 00700
2 SYIN_EOF BIT(1),  // END-OF-FILE FOR SYIN */ IR 00710
2 ENC_CMD Commands BIT(1), // END-OF-COMMANDS */ IR 00730
2 AREA_OVERFLOW BIT(1); // AREA OVERFLOW FLAG */ IR 00740
```

/* THIS DATA IS NEEDED FOR READING IN AND PARSGING COMMAND STRINGS. */

```c
11 DCL 1 IRCINFO EXTERNAL,
2 CSTEPPTR POINTER,  // TOP OF COMMAND STRING */ IR 00760
2 CSBSPTR POINTER,   // BOTTOM OF COMMAND STRING */ IR 00770
2 CSGLPTR POINTER,   // POINTER TO GARBAGE LIST */ IR 00780
2 CSGSRT PRINTER,    // POINTER TO CURRENT LINE */ IR 00800
2 CSTKPOS FIXED BINARY(15.0); // POSITION ON CURRENT LINE */ IR 00810
```

/* DATA NEEDED BY READLN AND WRITELN FOR ACCESSING THE TEXT FILE. */

```c
LAST_REGION AND END_LINE ARE INITIALIZED WITH DIR_REGION AND
DIR_LINE FROM THE DIRECTORY IF SETUP WAS NOT SPECIFIED AT THE
BEGINNING OF THE RUN. BEFORE THE DIRECTORIES ARE SAVED AT THE END,
DIR_REGION AND DIR_LINE ARE UPDATED WITH VALUES FROM HERE. */
```

```c
12 DCL 1 IRTXTEL EXTERNAL,
2 LAST_REGION FIXED BINARY(15.0); // HIGHEST REGION */ IR 00870
2 END_LINE FIXED BINARY(15.0);    // HIGHEST LINE # + 1 */ IR 00880
2 CUR_REGION FIXED BINARY(15.0);  // REG PRESENTLY IN CORE */ IR 00900
2 WRITE_FLAG BIT(1);  // IF SET, WRITE BLOCK */ IR 00910
```

/* THIS IS THE COMMAND TREE SETUP BY COMSET. */

```c
13 DCL COMTREE(51) FIXED BINARY(15.0) EXTERNAL;   // IR 00930
```

```c
14 */
```
/* IR -- INFORMATION RETRIEVAL SYSTEM */

STMT LEVEL NEST

15 1 DCL CASE(6) LABEL,
    /* BRANCH TABLE FOR CASE STMT */

    INSET CHAR(100) VARYING,
    /* SET UP PARAMETER */

    PAGE_No FIXED DEC(3,0) STATIC,
    /* PAGE NUMBER OF OUTPUT */

    (RUN DATE,RUN TIME) CHAR(8) STATIC,
    /* DATE AND TIME OF CURRENT RUN */

    I FIXED BIN(15,0) STATIC,
    /* LOOP COUNTER */

    IRMSG CHAR(62) STATIC INIT
    /* IR -- INFORMATION RETRIEVAL SYSTEM */;

    /* HEADER MESSAGE */
16 1 ON ENDFILE(SYSIN) SYSIN_EOF='1'B;
IR 01130
18 1 ON ENDFILE(POEM) POEM_EOF='1'B;
IR 01140

*/ PAGE HEADER */
IR 01150

20 1 ON ENDPAGE(SYSPRINT) BEGIN:
IR 01160
22 2 PUT PAGE_FILE(SYSPRINT) EDIT
IR 01170
(RUN_DATE,IRMSG,'PAGE',PAGE_NO)
(X(3),A,X(15),A,X(22),A,X(1),F(3));
IR 01190
23 2 PUT SKIP(2) FILE(SYSPRINT);
IR 01200
24 2 PAGE_NO = PAGE_NO+1;
IR 01210
25 2 END;
IR 01220

*/ CN AREA OVERFLOW PRINT WARNING, SET FLAG, WRITE DATA BASE TO DISK, AND EMPTY CORE RESIDENT AREAS */
IR 01230
IR 01240

26 1 CN AREA BEGIN;
IR 01250
28 2 AREA_OVERFLOW='1'B;
IR 01260
29 2 RUN_NO = C=RUN_NO;
IR 01270
30 2 DIR_REGION = LAST_REGION;
IR 01280
31 2 DIR_LINE = END_LINE;
IR 01290
32 2 OPEN FILE(CBASE) OUTPUT;
IR 01300
33 2 WRITE FILE(CBASE) FROM(DIRECTS);
IR 01310
34 2 PUT SKIP(3) FILE(SYSPRINT) EDIT
IR 01320
("*****---TERMINAL ERROR: DATA BASE OVERFLOW, DATA WRITTEN TO ' ,
DISK AS IS",")(X(5),A,A);
IR 01330
35 2 POEM_AREA,KEY_AREA=EMPTY;
IR 01340
36 2 IF WRITE_FLAG = '1'B THEN
IR 01350
37 2 WRITE FILE(TEXT) FROM(TEXTBUF) KEYFROM(CUR_REGION);
IR 01360
38 2 END;
IR 01370
IR 01380
/* INITIALIZE OUTPUT STRINGS AND OPEN PRINT FILE */

39 1 PAGE_NO = 1;
   IR 01390
40 1 RUN_DATE = DATE;
   IR 01400
41 1 RUN_DATE = SUBSTR(RUN_DATE,3,2) || ' ' || SUBSTR(RUN_DATE,5,2)
   IR 01410
42 1 || SUBSTR(RUN_DATE,1,2);
   IR 01420
43 1 RUN_TIME = SUBSTR(RUN_TIME,1,2) || ' ' || SUBSTR(RUN_TIME,3,2)
   IR 01430
44 1 || SUBSTR(RUN_TIME,5,2);
   IR 01440
45 1 OPEN FILE(SYSIN), FILE(SYSPRINT);
   IR 01450
46 1 /* INITIALIZE FLAGS AND SYSTEM PARMS */
   IR 01460
47 1 SYSIN_EOF, POEM_EOF, END_COMMANDS, AREA_OVERFLOW='0'B;
   IR 01470
48 1 HIGH_TEXT=5;
   IR 01500
49 1 LOW_NUMBER=20;
   IR 01510
50 1 /* INITIALIZE LINKED LIST FOR COMMAND STRING AND FILE I/O VLBS */
   IR 01520
51 1 ALLOCATE CSLIST_ELEMENT;
   IR 01530
52 1 CSTECPTR,CSTECPTR = CCSRUPTR;
   IR 01540
53 1 WGLPTR,CGLPTR = NULL;
   IR 01550
54 1 CUR_REGION = 0;
   IR 01560
55 1 WRITE_FLAG = '0'B;
   IR 01570
56 1 /* SET POINTERS TO POEM AND KEYLIST DIRECTORIES */
   IR 01580
57 1 FAPTR=ADDRI(POEM_AREA.Body);
   IR 01590
58 1 KAPTR=ADDRI(KEY_AREA_BODY);
   IR 01600
59 1 /* CHECK FOR SETUP PARAMETER IF FOUND CALL SETUP, ELSE READ DATA BASE FROM DISK */
   IR 01610
60 1 IF INSET='SETUP' THEN DO;
   IR 01620
61 1 RUN_NO = 1;
   IR 01630
62 1 CALL TITLE_PAGE;
   IR 01640
63 1 CALL SETUP;
   IR 01650
64 1 SIGNAL ENDPAGE(SYSPRINT);
   IR 01660
65 1 END;
   IR 01670
66 1 ELSE DO;
   IR 01680
67 1 OPEN FILE(DBASE) INPUT;
   IR 01690
68 1 READ FILE(DBASE) INTO(DIRECTS);
   IR 01700
69 1
/* IR -- INFORMATION RETRIEVAL SYSTEM */

STMT LEVEL NEST

/* MAIN PROCESSING LOOP: COMSET SETS UP COMMAND TREE, CASE ON
   FIRST ELEMENT AND CALL PROPER COMMAND PROCESSING ROUTINE */

72 1 CALL COMSET:
73 1 DC WHIL(~END_COMMANDS & ~AREA_OVERFLOW);
    /* ENTER CASE STMT */

74 1 1 GOTO CASE(COMTREE(I));
    /* CASE(1): ADD COMMAND */
75 1 1 CASE(1): CALL TRADD;
76 1 1 GOTO ENDCASE;

77 1 1 CASE(2): /* DELETE COMMAND */
78 1 1 CALL IRDRET;
    GOTO ENDCASE;

79 1 1 CASE(3): /* INITIALIZE COMMAND */
80 1 1 CALL TRADD;
    GOTO ENDCASE;

81 1 1 CASE(4): /* LIST COMMAND */
82 1 1 CALL IRLST;
    GOTO ENDCASE;

83 1 1 CASE(5): /* RETRIEVE COMMAND */
84 1 1 CALL IRDRET;
    GOTO ENDCASE;

85 1 1 CASE(6): /* SET COMMAND */
86 1 1 ENDCASE:
87 1 1 CALL COMSET;

88 1 1 END:

72 1 1 IF WRITE_FLAG = '1' THEN
73 1 1 WRITE_FILE(TEXT) FROM(TEXTBUF) KEYFROM(CUR_REGION);

74 1 1 /* WRITE DATA BASE TO DISK */
/* IR -- INFORMATION RETRIEVAL SYSTEM */

STMT LEVEL NEST

99 1 TITLE_PAGE: PROC;

/* THIS ROUTINE PRINTS THE TITLE PAGE */

100 2 PUT SKIP FILE(SYSPRINT) EDIT(('120)'**')(A);

101 2 DO I = 1 TO 25 BY 1:

102 2 1 PUT SKIP FILE(SYSPRINT) EDIT(('**','**')(A,X(118),A));

103 2 1 END;

104 2 PUT SKIP FILE(SYSPRINT) EDIT(('**','**')(A,X(26),A,X(30),A));

105 2 DO I = 1 TO 3 BY 1:

106 2 1 PUT SKIP FILE(SYSPRINT) EDIT(('**','**')(A,X(118),A));

107 2 1 END;

108 2 PUT SKIP FILE(SYSPRINT) EDIT

109 2 IF INSET = 'SETUP' THEN

110 2 PUT SKIP FILE(SYSPRINT) EDIT

111 2 ELSE

112 2 PUT SKIP FILE(SYSPRINT) EDIT

113 2 PUT SKIP FILE(SYSPRINT) EDIT

114 2 DO I = 1 TO 23 BY 1:

115 2 1 PUT SKIP FILE(SYSPRINT) EDIT(('**','**')(A,X(118),A));

116 2 1 END;

117 2 PUT SKIP FILE(SYSPRINT) EDIT(('120)'**')(A);

118 2 END TITLE_PAGE;

119 1 END IR;

IR 02180

IR 02190

IR 02200

IR 02210

IR 02220

IR 02230

IR 02240

IR 02250

IR 02260

IR 02270

IR 02280

IR 02290

IR 02300

IR 02310

IR 02320

IR 02330

IR 02340

IR 02350

IR 02360

IR 02370

IR 02380

IR 02390

IR 02400

IR 02410

IR 02420

IR 02430

IR 02440

IR 02450

IR 02460
<table>
<thead>
<tr>
<th>No.</th>
<th>Identifier</th>
<th>Attributes and References</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>AREA_OVERFLOW</td>
<td>IN IRFLAGS, STATIC, EXTERNAL, UNALIGNED, STRING(1), BIT 28, 45, 73, 90</td>
</tr>
<tr>
<td>15</td>
<td>CASE</td>
<td>(6) AUTOMATIC, INITIAL, LABEL 75, 77, 79, 81, 83, 85, 74</td>
</tr>
<tr>
<td>14</td>
<td>CLE_CARD</td>
<td>IN CSLIST_ELEMENT, BASED(CSCURPTR), UNALIGNED, STRING(80), ()</td>
</tr>
<tr>
<td>2</td>
<td>COMSET</td>
<td>EXTERNAL, ENTRY, DECIMAL, FLOAT(SINGLE) 72, 86</td>
</tr>
<tr>
<td>13</td>
<td>COMTREE</td>
<td>(5) STATIC, EXTERNAL, ALIGNED, BINARY, FIXED(15, 0) 74</td>
</tr>
<tr>
<td>11</td>
<td>CSPOIPTR</td>
<td>IN IRCINFO, STATIC, EXTERNAL, ALIGNED, POINTER 49</td>
</tr>
<tr>
<td>11</td>
<td>CSCURPTR</td>
<td>IN IRCINFO, STATIC, EXTERNAL, ALIGNED, POINTER 48, 49</td>
</tr>
<tr>
<td>11</td>
<td>CSGLPTR</td>
<td>IN IRCINFO, STATIC, EXTERNAL, ALIGNED, POINTER 50</td>
</tr>
<tr>
<td>14</td>
<td>CSLIST_ELEMENT</td>
<td>BASED(CSCURPTR), STRUCTURE 48</td>
</tr>
<tr>
<td>11</td>
<td>CSTOKPOS</td>
<td>IN IRCINFO, STATIC, EXTERNAL, ALIGNED, BINARY, FIXED(15, 0) 49</td>
</tr>
<tr>
<td>11</td>
<td>CSIEPPTR</td>
<td>IN IRCINFO, STATIC, EXTERNAL, ALIGNED, POINTER 49</td>
</tr>
<tr>
<td>12</td>
<td>CUR_REGION</td>
<td>IN IRTEXTL, STATIC, EXTERNAL, ALIGNED, BINARY, FIXED(15, 0) 37, 51, 89</td>
</tr>
<tr>
<td></td>
<td>DATE</td>
<td>BUILT-IN FUNCTION 40</td>
</tr>
<tr>
<td>STATEMENT NO.</td>
<td>IDENTIFIER</td>
<td>LENGTH IN BYTES</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>15</td>
<td>CASE</td>
<td>48</td>
</tr>
<tr>
<td>13</td>
<td>COMTREE</td>
<td>10</td>
</tr>
<tr>
<td>14</td>
<td>CSLIST_ELEMENT</td>
<td>84</td>
</tr>
<tr>
<td>5</td>
<td>DIRECTS</td>
<td>28158</td>
</tr>
<tr>
<td>11</td>
<td>IRCINFO</td>
<td>18</td>
</tr>
<tr>
<td>10</td>
<td>IRFLAGS</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>IRPARMS</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>IRTXTBL</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>TEXT_LINE</td>
<td>1200</td>
</tr>
</tbody>
</table>
STORAGE REQUIREMENTS.

THE STORAGE AREA FOR THE PROCEDURE LABELLED IR IS 508 BYTES LONG.

THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 16 IS 196 BYTES LONG.

THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 18 IS 196 BYTES LONG.

THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 21 IS 196 BYTES LONG.

THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 27 IS 288 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED TITLE_PAGE IS 224 BYTES LONG.

THE PROGRAM CSECT IS NAMED IR AND IS 4250 BYTES LONG.

THE STATIC CSECT IS NAMED *****IRA AND IS 1936 BYTES LONG.

*STATISTICS* SOURCE RECORDS = 246, PROG TEXT STMTS = 119, OBJECT BYTES = 4250
<table>
<thead>
<tr>
<th>TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFSET (HEX) 0000 0054 005C</td>
</tr>
<tr>
<td>STATEMENT NO  17   17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFSET (HEX) 00CC 0054 005C</td>
</tr>
<tr>
<td>STATEMENT NO  19   19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFSET (HEX) 0000 0048 0CF8 0108 010E</td>
</tr>
<tr>
<td>STATEMENT NO  21   22   23   24   25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFSET (HEX) 0000 00A8 00B0 008E 00CC 00DA 00E4 00EE 0140 0164 0184 01CC</td>
</tr>
<tr>
<td>STATEMENT NO  27   28   29   30   31   32   33   34   35   36   37   38</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFSET (HEX) 0000 0044 0072 007A 00CC 00E0 0156 015E 0180 01C4 025E 0282 02FC 038C 0426 04CC 051A 052E</td>
</tr>
<tr>
<td>STATEMENT NO  99   100   101   102   103   104   105   106   107   108   109   110   111   112   113   114   115   116   117</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFSET (HEX) 0000 029C 02AA 02B8 02C6 02DA 02DC 02FC 032C 0344 0374 037E 039E 03A8 03B2 03D2 03E6 03FA 0404</td>
</tr>
<tr>
<td>STATEMENT NO  1     16   18   20   22   26   39   40   41   42   43   44   45   46   47   48   49   50   51   52</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFSET (HEX) 042C 044C 0456 0460 046A 0480 0484 0488 0498 04A2 04B0 04B4 04BE 04DE 04EF 04E9 04F2</td>
</tr>
<tr>
<td>STATEMENT NO  55   56   57   58   59   60   61   62   63   64   65   66   67   68   69   70   71   72   73</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFSET (HEX) 05B2 05B6 05C0 0594 059E 05A2 05AC 05B0 05BA 05BE 05C8 05D2 05D6 05F6 05E3 0574 0574 057E 0580</td>
</tr>
<tr>
<td>STATEMENT NO  76   77   78   79   80   81   82   83   84   85   86   87   88   89   90   91   92   93   94</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFSET (HEX) 06AE 06AE 07C0</td>
</tr>
<tr>
<td>STATEMENT NO  97   98   119</td>
</tr>
</tbody>
</table>
COMPILER DIAGNOSTICS.

WARNINGS.

IEM07641  ONE OR MORE FIXED BINARY ITEMS OF PRECISION 15 OR LESS HAVE BEEN GIVEN HALFWORD STORAGE.  ARE FLAGGED "*****" IN THE XREF/ATR LIST.

END OF DIAGNOSTICS.

AUXILIARY STORAGE WILL NOT BE USED FOR DICTIONARY WHEN SIZE = 120K

COMPILE TIME  .12 MINS
FLIPPED TIME  .66 MINS