S E T L D A T A S T R U C T U R E S

SETL NEWSLETTER NUMBER 189

R. DEWAR
A. GRAND
E. SCHONBERG
J. SCHWARTZ

JULY 7TH 1977

THIS NEWSLETTER DESCRIBES THE DATA STRUCTURES WHICH WILL BE USED FOR THE RUN TIME SYSTEMS OF BOTH THE INTERPRETED AND COMPILED VERSIONS OF THE NEW SETL SYSTEM.

THE INTRODUCTION OF THE NOTION OF BASINGS HAS PROVIDED AN OPPORTUNITY FOR REDESIGN OF THE SETL RUNTIME SYSTEM SINCE IT WILL BE NECESSARY TO MODIFY MUCH OF THE RUN TIME LIBRARY IN ANY CASE TO SUPPORT THE BASINGS.

CONSEQUENTLY, THE DATA STRUCTURES HAVE BEEN COMPLETELY REDESIGNED AND IT IS EXPECTED THAT PROGRAMS WILL RUN FASTER WITH THE REDESIGNED STRUCTURES EVEN IF THE BASING DECLARATIONS ARE NOT USED.

TABLE OF CONTENTS

TABLE OF CONTENTS

1. FORM OF STORAGE

AS IN THE PREVIOUS SYSTEM, STORAGE IS ARRANGED AS A CONTIGUOUS SEQUENCE OF 'WORDS'. EACH SETL WORD MAY BE COMPOSED OF ONE OR MORE MACHINE WORDS DEPENDING ON THE MACHINE WORD SIZE.

THE SETL WORD (HENCEFORTH ALWAYS CALLED SIMPLY A WORD) IS CAPABLE OF HOLDING A POINTER FIELD WITH 7 BITS LEFT OVER. IF MORE BITS ARE AVAILABLE, THEY CAN BE MADE USE OF IN VARIOUS SITUATIONS.

THERE ARE BASICALLY TWO FORMATS FOR WORDS:

VALUE SPECIFIER: A ONE WORD QUANTITY USED TO HOLD (DIRECTLY OR INDIRECTLY) A SETL DATA VALUE. THESE CORRESPOND TO THE ROOT WORDS OF THE OLD SYSTEM.

DATA WORD: IN SOME CASES, VALUE SPECIFIERS ARE USED TO POINT TO DATA BLOCKS CONSISTING OF ONE OR MORE DATA WORDS IN DATATYPE DEPENDENT FORMAT.

1.1. VALUE SPECIFIER

A VALUE SPECIFIES A DATA VALUE IN THE SETL SYSTEM AND HAS THE FOLLOWING UNIFORM FIELDS:

IS←NIL: ON FOR SPECIFIER FOR NIL VALUE (2.7.)

TYPE: A FIELD GIVING THE TYPE CODE OF THE VALUE. THIS FIELD MUST BE AT LEAST 5 BITS LONG, BUT MAY BE LONGER IF CONVENIENT FOR EFFICIENT ACCESS. IT CONTAINS A SYSTEM CONSTANT VALUE (WHOSE NAME IS OF THE FORM T←XXX) WHICH INDICATES THE TYPE.

VALUE: THIS FIELD CONTAINS EITHER THE DATA VALUE OR A POINTER TO A DATA BLOCK IN THE HEAP WHICH CONTAINS THE VALUE. THE TYPE CODE (IN THE TYPE FIELD) INDICATES WHICH OF THESE TWO FORMS IS USED AND THE EXACT SIGNIFICANCE OF THE VALUE FIELD.

IS←SHARED: A FLAG WHICH IS SET IN A SPECIFIER TO INDICATE THAT THE VALUE REFERRED TO IS SHARED AND THAT THE VALUE MUST BE COPIED BEFORE IT IS MODIFIED.

IS←MULTI: A FLAG USED IN MAP FORMATS TO INDICATE THAT AN ENTRY IS FOR A RANGE SET (RATHER THAN A SINGLE RANGE VALUE).

THE IS←NIL, TYPE AND VALUE FIELDS ARE ADJACENT AND IN THAT ORDER. THE FOLLOWING COMPOSITE FIELDS ARE DEFINED:

NTYPE: IS←NIL + TYPE
TVALUE: TYPE + VALUE
NTVALUE: IS←NIL + TYPE + VALUE

1.2. DATA WORDS

CERTAIN DATA VALUES (LARGE PRIMITIVE VALUES AND ALL COMPOSITE VALUES) CANNOT BE REPRESENTED BY A SINGLE VALUE SPECIFIER.

THESE VALUES ARE REPRESENTED BY ONE OR MORE DATA WORDS WHICH ARE POINTED TO BY THE VALUE FIELD OF THE SPECIFIER. SUCH A COLLECTION OF DATA WORDS IS CALLED A DATA BLOCK.

EACH DATA BLOCK IS RECARDED AS AN ENTITY BY THE GARBAGE COLLECTOR AND THERE IS A STANDARD HEADER AT THE BEGINNING OF EACH DATA BLOCK FOR USE BY THE GARBAGE COLLECTOR. THIS HEADER HAS THE FOLLOWING FIELDS:

HEDRTYPE: A UNIQUE CODE FOR THE TYPE OF DATA BLOCK (H←XXX)
GLINK: POINTER FIELD FOR USE BY THE GARBAGE COLLECTOR

AS WITH ALL FIELDS IN DATA BLOCKS, THE POSITION OF THE FIELDS (WORD AND BIT OFFSET) IS A FUNCTION OF THE IMPLEMENTATION, SUBJECT TO THE REQUIREMENT THAT ALL POINTERS HAVE THE SAME POSITION IN A WORD (I.E. THEY CORRESPOND TO THE STANDARD FIELD STD←PTR IN POSITION).

GLINK IS USED TO BUILD BACK CHAINS BY THE GARBAGE COLLECTOR. SEE GARBAGE COLLECTOR DESCRIPTION FOR FULL DETAILS. BETWEEN GARBAGE COLLECTIONS, IT MAY CONTAIN ANY VALUE IN THE RANGE (0 - SYM←START) AND THE GARBAGE COLLECTOR PRESERVES THIS VALUE (SYM←START IS THE LOWEST POSSIBLE REAL HEAP ADDRESS, AND THUS VALUES IN THIS RANGE ARE DISTINGUISHABLE FROM HEAP POINTERS).

DURING THE GARBAGE COLLECTION PROCESS, AN ADDITIONAL FIELD CALLED GSIZE IS USED IN DEAD BLOCKS ONLY. SINCE THIS FIELD IS NOT NEEDED IN ACTIVE BLOCKS, IT MAY OVERLAP OTHER FIELDS (BUT THERE MUST BE ROOM FOR IT!).

THE FORMAT OF THE REMAINING DATA WORDS IN THE DATA BLOCK IS DEPENDENT ON THE DATATYPE INVOLVED, AS INDICATED BY THE TYPE FIELD OF THE REFERENCING SPECIFIER (AND THE VALUE IN HEDRTYPE). IF MORE THAN ONE DATA WORD IS INVOLVED, THEN THERE IS TYPICALLY ADDITIONAL HEADER INFORMATION WHICH (AMONG OTHER INFORMATION) GIVES OR IMPLIES THE LENGTH OF THE BLOCK.

THE DETAILED FORMAT OF DATA WORDS IS EXPLAINED DATATYPE BY DATATYPE IN THE SUCCEEDING SECTIONS.

2. TYPED PRIMITIVE DATA

THIS SECTION CONTAINS DETAILS ON THE FORMAT OF PRIMITIVE DATA ITEMS WHICH DO NOT CONTAIN COMPONENT VALUES (I.E. ALL TYPES EXCEPT SETS AND TUPLES).

EVERY DATA VALUE IN THE SETL SYSTEM IS REPRESENTED BY A VALUE SPECIFIER (OFTEN CALLED JUST A SPECIFIER). IN SOME CASES (SHORT ITEMS), THE DATA VALUE CAN BE COMPLETELY CONTAINED IN THE SPECIFIER. IN OTHER CASES (LONG ITEMS), THE VALUE FIELD OF THE SPECIFIER CONTAINS A POINTER TO DATA WORDS WHICH DESCRIBE THE DATA VALUE.

NOTE THAT ONLY THE TYPE AND VALUE FIELDS CONTAIN OR REPRESENT THE DATA VALUE ITSELF. ALL THE OTHER FIELDS IN A SPECIFIER ARE INDEPENDENT OF THE DATA VALUE AND THEIR USE DEPENDS ON THE CONTEXT IN WHICH THE SPECIFIER APPEARS.

IN CASES WHERE THERE ARE LONG AND SHORT FORMS FOR THE SAME DATATYPE, THEN ITEMS ARE REPRESENTED IN THE SHORT FORM IF POSSIBLE, BUT THIS IS NOT REQUIRED. FOR EXAMPLE, A ONE CHARACTER STRING COULD BE STORED IN EITHER MANNER, AND THE RESULT WOULD BE INDISTINGUISHABLE TO THE PROGRAM (FOR EXAMPLE, THEY WOULD APPEAR TO BE EQUAL).

2.1. INTEGER

AN INTEGER IN SETL IS A SIGNED INTEGRAL VALUE WITH NO LIMIT ON THE MAGNITUDE OTHER THAN THAT IMPOSED BY MEMORY CONSTRAINTS. THERE ARE TWO FORMATS FOR INTEGERS, CALLED SHORT INTEGER AND LONG INTEGER.

2.1.1. SHORT INTEGER

SHORT INTEGERS RANGE FROM 0-MAXSI, WHERE MAXSI IS THE LARGEST INTEGER WHICH WILL FIT INTO THE VALUE FIELD (I.E. ALL BITS OF THE VALUE FIELD SET ON).

THE SPECIFIER HAS:

TYPE:	T←INT
VALUE:	INTEGER VALUE

THE TYPE FIELD VALUE MUST BE ZERO. THIS PARTICULAR VALUE IS SIGNIFICANT IN THAT IT ALLOWS FOR RAPID CODING OF THE INTEGER ADDITION FUNCTION.

THE INTEGER VALUE RANGES FROM 0 TO THE IMPLEMENTATION CONSTANT MAXSI.

ONLY SMALL POSITIVE INTEGERS CAN BE STORED IN THIS MANNER. LARGE AND NEGATIVE INTEGERS ARE STORED IN LONG INTEGER FORMAT.

2.1.2. LONG INTEGER

THE SPECIFIER FOR A LONG INTEGER HAS:

TYPE: T.LINT
VALUE: POINTS TO AN INTEGER DATA BLOCK

THE FIRST WORD OF AN INTEGER DATA BLOCK HAS:

LI.NWORDS: NUMBER OF WORDS IN BLOCK INCLUDING 1ST WORD

REMAINING WORDS OF INTEGER DATA BLOCKS ARE IN A FORMAT WHICH IS CONVENIENT TO THE PARTICULAR MACHINE AND IMPLEMENTATION. THE DETAILS OF THIS FORMAT ARE OF SIGNIFICANCE ONLY TO THE LONG INTEGER ROUTINES AND ARE DESCRIBED IN THESE ROUTINES.

2.2. STRING

A STRING VALUE IN SETL IS AN ARBITRARY LENGTH SEQUENCE OF CHARACTERS. THE EXACT RANGE OF POSSIBLE CHARACTERS DEPENDS ON THE IMPLEMENTATION ENVIRONMENT, BUT SHOULD INCLUDE UPPER/LOWER CASE LETTERS IF POSSIBLE. THERE ARE TWO FORMATS FOR STRING VALUES:

2.2.1. SHORT STRING

SHORT STRING FORMAT CAN REPRESENT STRINGS FROM 0 (THE NULL STRING) TO STRINGS OF LENGTH SC.MAX (AN IMPLEMENTATION DEPENDANT SYSTEM CONSTANT).

THE SPECIFIER FOR A SHORT STRING VALUE HAS:

TYPE: T.STRING
VALUE: SUBDIVIDED INTO TWO FIELDS AS FOLLOWS:

SC.NCHARS: NUMBER OF CHARACTERS IN STRING
SC.STRING: STRING CHARACTERS

THE CHARACTERS ARE STORED LEFT JUSTIFIED, RIGHT FILLED. THE FILL CHARACTER IS BINARY ZERO BITS. THE LENGTH OF A SINGLE CHARACTER IN BITS IS GIVEN BY THE SYSTEM CONSTANT CH.SIZ.

2.2.2. LONG STRING

THERE ARE TWO METHODS OF STORING LONG STRING VALUES. THE CHOICE BETWEEN THESE METHODS DEPENDS ON THE IMPLEMENTATION WORD CAPACITY. THE LIBRARY CODE HAS AN ASSEMBLY SWITCH (SSI) WHICH DETERMINES THE CHOICE FOR A GIVEN MACHINE.

2.2.2.1. DIRECT FORMAT LONG STRING

THIS FORMAT CORRESPONDS TO SSI BEING SET OFF AND IS USED IF A STRING DESCRIPTOR AS DESCRIBED HERE CAN FIT INTO THE VALUE FIELD OF A SPECIFIER. THE SPECIFIER FOR A DIRECT FORMAT LONG STRING VALUE HAS:

```
TYPE:     T←ICHAR
VALUE:    STRING DESCRIPTOR
```

THE STRING DESCRIPTOR CONSISTS OF THE FOLLOWING THREE SUBFIELDS:

```
LC←LEN:   LENGTH OF STRING IN CHARACTERS
LC←OFS:   OFFSET TO FIRST CHARACTER IN WORD
LC←PTR:   POINTER TO LONG STRING DATA BLOCK
```

THE DATA BLOCK WHICH CONTAINS THE ACTUAL STRING CHARACTERS IS CALLED A LONG STRING DATA BLOCK AND HAS THE FOLLOWING FIELDS:

```
HEDRTYPE:   H←LCHARS
CLINK:      NOT USED (0)
LC←NWORDS:  NUMBER OF WORDS IN DATA BLOCK, INCLUDING HEADER
```

THE CHARACTERS ARE STORED IN THE REMAINING AREA OF THE BLOCK IN A FORMAT WHICH IS IMPLEMENTATION DEPENDANT AND DESCRIBED BY THE CODING OF THE STRING MANIPULATION ROUTINES.

NOTE THAT THE STRING DESCRIPTOR MAY REFERENCE ONLY A SUBSTRING OF THE STRING CONTAINED IN A LONG STRING DATA BLOCK.

2.2.2.2. INDIRECT FORMAT LONG STRING

THIS FORMAT CORRESPONDS TO A SSI BEING SET AND IS USED IF THE THREE FIELDS NEEDED TO DESCRIBE A STRING CANNOT FIT INTO THE VALUE FIELD OF A SPECIFIER. THE SPECIFIER FOR AN INDIRECT FORMAT LONG STRING VALUE HAS:

```
TYPE:     T←ICHAR
VALUE:    STRING DESCRIPTOR
```

THE STRING DESCRIPTOR IS A POINTER TO A INDIRECT STRING DATA BLOCK WHICH HAS THE FOLLOWING FORMAT:

```
HEDRTYPE:   H←IC
```

GLINK: NOT USED (0)
LC←LEN: LENGTH OF STRING IN CHARACTERS
LC←OFS: OFFSET TO FIRST CHARACTER IN WORD
LC←PTR: POINTER TO LONG STRING DATA BLOCK

THE DATA BLOCK WHICH CONTAINS THE ACTUAL STRING CHARACTERS IS CALLED A LONG STRING DATA BLOCK AND HAS THE FOLLOWING FIELDS:

HEDRTYPE: H←LCHARS
CLINK: NOT USED (0)
LC←NWORDS: NUMBER OF WORDS IN DATA BLOCK, INCLUDING HEADER

THE CHARACTERS ARE STORED IN THE REMAINING AREA OF THE BLOCK IN A FORMAT WHICH IS IMPLEMENTATION DEPENDANT AND DESCRIBED BY THE CODING OF THE STRING MANIPULATION ROUTINES.

NOTE THAT THE INDIRECT STRING DATA BLOCK MAY REFERENCE ONLY A SUBSTRING OF THE STRING CONTAINED IN A LONG STRING DATA BLOCK.

2.3. REAL

THERE IS NO PROVISION FOR SHORT REAL VALUES SINCE IT IS ASSUMED THAT THE VALUE FIELD IS TOO SHORT TO CONTAIN A MEANINGFUL REAL VALUE. HOWEVER, IT IS ASSUMED THAT A FULL SETL WORD WILL HOLD A REAL VALUE.

THE SPECIFIER FOR A LONG REAL VALUE HAS:

TYPE: T←REAL
VALUE: POINTER TO REAL DATA BLOCK

A REAL DATA BLOCK CONTAINS THE FOLLOWING FIELDS:

HEDRTYPE: H←REAL
CLINK: NOT USED (0)
RVAL: VALUE OF REAL

THE SIZE OF A REAL DATA BLOCK IS GIVEN BY THE SYSTEM CONSTANT RNV.

2.4. SUBROUTINE

THE SPECIFIER FOR A SUBROUTINE VALUE HAS:

TYPE: T←SUBR
VALUE: CODE POINTER FOR SUBROUTINE

THE FORMAT OF THE CODE POINTER DEPENDS ON THE FORM OF THE EXECUTING PROGRAM. FOR THE INTERPRETIVE VERSION, IT IS A QUADRUPLE POINTER. FOR THE COMPILED VERSION, IT IS A DIRECT CODE POINTER.

2.5. FUNCTION

THE SPECIFIER FOR A FUNCTION VALUE HAS:

TYPE: T←FUNC
VALUE: CODE POINTER FOR FUNCTION

THE FORMAT OF THE CODE POINTER DEPENDS ON THE FORM OF THE EXECUTING PROGRAM. FOR THE INTERPRETIVE VERSION, IT IS A QUADRUPLE POINTER. FOR THE COMPILED VERSION, IT IS A DIRECT CODE POINTER.

2.6. ATOM

AN ATOM IN SETL REPRESENTS A UNIQUE VALUE WHICH CAN BE COMPARED FOR EQUALITY BUT NOT OTHERWISE MANIPULATED. THERE ARE TWO FORMS FOR ATOMS, SHORT AND LONG.

2.6.1. SHORT ATOM

THE SPECIFIER FOR A SHORT ATOM VALUE HAS:

TYPE: T←ATOM
VALUE: IDENTIFIES ATOM VALUE

THERE ARE TWO POSSIBILITIES FOR VALUE:

FOR A NAMED CONSTANT ATOM (E.G. TRUE), VALUE IS A POINTER TO THE SYMBOL TABLE ENTRY FOR THE NAME. THIS VALUE IS ALWAYS IN THE RANGE 1 TO SYM←LEN.

FOR ALL OTHER SHORT ATOM VALUES, VALUE IS SIMPLY A UNIQUE IDENTIFYING INTEGER IN THE RANGE SYM←LEN+1 TO MAXSI. NEWAT RETURNS A NEW BLANK ATOM BY INCREMENTING A COUNTER. IF THE COUNTER OVERFLOWS, THEN NEWAT RETURNS LONG BLANK ATOMS FROM THEN ON.

2.6.2. LONG ATOM

LONG ATOM VALUES ARE USED INSTEAD OF SHORT ATOMS IN THE FOLLOWING CASES:

1) THE COUNTER FOR SHORT ATOMS HAS OVERFLOWED
2) THE ATOM IS IN A PLEX BASE (4.3.2.5.)

THE SPECIFIER FOR A LONG ATOM VALUE IS:

TYPE: T←LATOM
VALUE: POINTER TO ATOM DATA BLOCK

THE ATOM DATA BLOCK CONTAINS THE FOLLOWING FIELDS:

HEDRTYPE: H←LATOM
GLINK: NOT USED (0)
LA←VALUE: IDENTIFIES ATOM, SEE BELOW
LA←NWORDS: LENGTH OF DATA BLOCK
LA←NLMAPS: USED FOR PLEX BASES (4.3.2.5.)

THERE ARE TWO POSSIBILITIES FOR LA←VALUE:

FOR A NAMED CONSTANT ATOM (E.G. TRUE), LA←VALUE IS A POINTER TO THE SYMBOL TABLE ENTRY FOR THE NAME. THIS VALUE IS ALWAYS IN THE RANGE 1 TO SYM←LEN.

FOR ALL OTHER LONG ATOM VALUES, LA←VALUE IS SIMPLY A UNIQUE IDENTIFYING INTEGER GREATER THAN MAXSI. IT IS REQUIRED THAT THIS FIELD BE LONG ENOUGH SO THAT THE QUESTION OF OVERFLOW DOES NOT ARISE.

THE LA←NLMAPS IS USED ONLY FOR PLEX BASES AS DESCRIBED IN SECTION 4.3.2.5. AND IS 0 FOR ALL OTHER CASES.

2.7. NIL VALUE

THE NIL VALUE IS TREATED SPECIALLY. ITS SPECIFIER HAS A TYPE AND VALUE WHICH CORRESPOND TO SOME PROPER DEFINED VALUE (OF THE APPROPRIATE TYPE IN THE CASE OF AN OBJECT WITH A REPR). THE IS←NIL BIT OF THIS SPECIFIER IS SET TO INDICATE THAT THE VALUE IS NIL.

FROM THE SETL LANGUAGE POINT OF VIEW THERE IS ONLY ONE NIL VALUE. THE USE OF MULTIPLE REPRESENTATIONS OF NIL WITHIN THE LIBRARY IS USEFUL IN THE CASE OF BASED MAP AND SET FORMATS, WHERE NIL VALUES CAN RETAIN TYPING INFORMATION.

AIN ADDITION, THE FACT THAT NIL VALUES APPEAR TO HAVE A PROPER VALE OF THE CORRECT TYPE MEANS THAT CODE WHICH OMITS THE IS←NIL CHECK ALWAYS PRODUCES RESULTS WHICH, THOUGH THEY MAY NOT BE CORRECT, DO NOT RESULT IN FAILURE OF SYSTEM INTEGRITY.

2.8. BOOLEAN VALUES

IN SETL, TRUE IS A DISTINGUISHED CONSTANT ATOM VALUE, AND IS STORED IN THE SAME MANNER AS ANY OTHER LONG ATOM VALUE.

FALSE IS REPRESENTED BY THE NIL VALUE. THE PREDEFINED SYSTEM IDENTIFIER FALSE HAS NIL AS ITS VALUE.

3. UNTYPED PRIMITIVE DATA

IN ADDITION TO THE TYPED DATA STRUCTURES DESCRIBED IN THE PREVIOUS SECTION, THERE EXISTS UNTYPED DATA WHICH DOES NOT CARRY A TYPE CODE.

UNLIKE ANY OTHER DATA IN THE SYSTEM, SUCH VALUES CANNOT BE IDENTIFIED BY THE BIT PATTERN OF THEIR SPECIFIER.

TO MAINTAIN THE INTEGRITY OF THE ENVIRONMENT SO THAT THE GARBAGE COLLECTOR CAN OPERATE CORRECTLY, THE MANNER IN WHICH UNTYPED DATA CAN APPEAR IS RESTRICTED AS DESCRIBED IN THIS SECTION.

3.1. UNTYPED INTEGER

AN UNTYPED INTEGER FITS INTO ALL OR PART OF A SETL WORD. IT TYPICALLY CORRESPONDS TO A SIGNED INTEGER VALUE WHICH IS THE HARDWARE INTEGER SIZE.

3.2. NIL UNTYPED INTEGER

THE NIL UNTYPED INTEGER VALUE IS REPRESENTED BY A UNIQUE MACHINE DEPENDANT BIT PATTERNS WHICH MEETS THE REQUIREMENT THAT IT IS NEVER PRODUCED AS THE RESULT OF ANY INTEGER OPERATION ON DEFINED VALUES WHERE THE RESULT IS PROPERLY DEFINED AND IN RANGE.

TYPICAL CHOICES ARE NEGATIVE ZERO (1'S COMPLEMENT) OR THE LARGEST NEGATIVE NUMBER (2'S COMPLEMENT).

3.3. UNTYPED REAL

AN UNTYPED REAL FITS INTO ALL OR PART OF A SETL WORD. IT TYPICALLY CORRESPONDS TO A SIGNED REAL VALUE WHICH IS THE HARDWARE REAL SIZE.

THE FORMAT OF AN UNTYPED REAL IS THE SAME AS THE FORMAT OF A REAL DATA WORD (SEE SECTION 3.4).

3.4. NIL UNTYPED REAL

THE NIL UNTYPED REAL VALUE IS REPRESENTED BY A UNIQUE MACHINE DEPENDANT BIT PATTERN WHICH MEETS THE REQUIREMENT THAT IT IS NEVER PRODUCED AS THE RESULT OF ANY REAL OPERATION ON DEFINED VALUESWHERE THE RESULT IS PROPERLY DEFINED.

WHERE AN OPERATION ON REAL VALUES PRODUCES AN IMPROPER RESULT (E.G. 0.0/0.0), THE NIL VALUE MUST BE GENERATED.

WHERE A REAL OPERATION OPERATES ON NIL REAL VALUES, IT IS DESIRABLE TO GIVE AN ERROR, BUT THIS IS NOT REQUIRED.

A TYPICAL CHOICE FOR THE NIL REAL VALUE IS AN UNNORMALIZED REAL VALUE.

3.5. SKIP WORD

TO PREVENT THE GARBAGE COLLECTOR FROM PROCESSING UNTYPED DATA, A SPECIAL DUMMY SPECIFIER PRECEDES SUCH WORDS IN ANY CONTEXT WHERE THE GARBAGE COLLECTOR SCANS A VECTOR OF SPECIFIERS IN SEQUENCE (E.G. IN THE SYMBOL TABLE AND IN THE STACK). THE SPECIFIER FORMAT IS:

TYPE:	T←SKIP
VALUE:	NUMBER OF FOLLOWING SPECIFIERS TO BE SKIPPED

4. NON-PRIMITIVE DATA

NON-PRIMITIVE DATA INCLUDES ALL SET AND TUPLE FORMATS. SUCH VALUES ARE REPRESENTED BY A SPECIFIER WHICH HAS:

TYPE:	INDICATES TUPLE OR SET TYPE
VALUE:	POINTER TO SET OR TUPLE DATA BLOCK

THE FORMAT OF THE DATA BLOCK DEPENDS ON THE PARTICULAR TYPE OF TUPLE OR SET INVOLVED, BUT CERTAIN FIELDS ARE COMMON TO ALL (OR MOST) FORMATS:

HEDRTYPE:	INDICATES TUPLE OR SET TYPE
GLINK:	NOT USED (0)
NELT:	NUMBER OF ELEMENTS (IF IS←NELTOK IS SET)
IS←NELTOK:	FLAG SET ON IF NELT CONTAINS A VALID VALUE
HASH:	HASH CODE (IF IS←HASHOK IS SET)
IS←HASHOK:	FLAG SET ON IF HASH CONTAINS A VALID VALUE
BASEA:	POINTER TO BASE ARRAY DATA BLOCK
BASEO:	OFFSET TO FIRST WORD IN BASE ARRAY DATA BLOCK
FORM:	INDEX TO CORRESPONDING ENTRY IN FORM TABLE (7.)

NOTE THAT THE HEDRTYPE VALUE CONTAINS MORE SPECIFIC INFORMATION THAN THE TYPE FIELD OF THE SPECIFIER. I.E. THERE ARE CASES IN WHICH MORE THAN ONE DIFFERENT TYPE OF OBJECT HAVE THE SAME SPECIFIER TYPE, BUT THE DIFFERENCES ARE RESOLVED BY THE HEDRTYPE VALUE.

IF THE TUPLE OR SET IS OF A BASED FORM, THE BASEA AND BASEO FIELDS POINT TO A BASE ARRAY, AS DESCRIBED IN THE FOLLOWING SECTION. THE FORM TABLE ENTRY INDEXED BY FORM INDICATES THE NUMBER OF ENTRIES IN THIS BASE ARRAY.

4.1. BASE ARRAY

TUPLE AND SET HEADERS CONTAIN POINTERS TO A BASE ARRAY IN THE CASE WHERE THE CORRESPONDING OBJECT CONTAINS BASED SUBOBJECTS. THE NUMBER AND ORDER OF ENTRIES IN THIS ARRAY ALWAYS CORRESPONDS TO THE STRUCTURE IN THE RELATED FORM TABLE ENTRY AS INDICATED BY THE SETTING OF THE FORM FIELD IN THE HEADER.

A BASE ARRAY DATA BLOCK HAS THE FOLLOWING FORMAT:

HEDRTYPE:	H←BASEA
GLINK:	NOT USED (0)
BA←NWORDS:	LENGTH IN WORDS (INCLUDING HEADER)

THE REMAINING WORDS OF THE DATA BLOCK ARE STANDARD FORMAT SPECIFIERS FOR THE RELEVANT BASE SETS.

NOTE THAT A GIVEN SET OR TUPLE HEADER MAY REFERENCE ONLY A CONTIGUOUS SUBSECTION OF THE ARRAY.

THE ORDER OF BASES IN THE BASE ARRAY IS THE SAME AS THE ORDER IN WHICH THE BASES ARE MENTIONED IN THE CORRESPONDING REPR.

4.2. TUPLE FORMAT

IN ADDITION TO THE STANDARD FIELDS FOR NON-PRIMITIVE OBJECTS, ALL TUPLE DATA BLOCKS HAVE ONE STANDARD FIELD:

MAXINDX: INDEX OF THE LAST COMPONENT ALLOCATED
IS←RANGE: USED ONLY IN MAP ITERATORS (6.3.)

NOTE THAT MAXINDX MAY BE DIFFERENT FROM THE VALUE IMPLIED BY THE CARDINALITY (NELT) SINCE IT IS USUAL TO ALLOCATE A GROWTH SPACE FOR TUPLE VALUES.

4.2.1. STANDARD TUPLE

A STANDARD FORMAT TUPLE IS REPRESENTED BY A SPECIFIER WHICH HAS:

TYPE: T←TUPLE
VALUE: POINTER TO TUPLE DATA BLOCK

THE TUPLE DATA BLOCK HAS THE STANDARD FIELDS, WITH

HEDRTYPE: H←TUPLE

FOLLOWING THE HEADER IS A ZERO-ORIGIN VECTOR OF DATA VALUES. THESE ARE ORDINARY VALUE SPECIFIERS GIVING THE VALUES OF THE NELT SUCCESSIVE ELEMENTS OF THE TUPLE, OR THE NIL VALUE FOR INDICES FOR WHICH THE TUPLE VALUE IS NOT DEFINED. NOTE THAT THE ZERO ENTRY ALWAYS CONTAINS NIL AND CANNOT BE MODIFIED (SINCE TUPLES IN THE SETL LANGUAGE ARE ONES ORIGIN). THE EXTRA ZERO INDEX VALUE SPEEDS INDEXED REFERENCES.

MAXINDX IMPLIES THE NUMBER OF WORDS ALLOCATED TO THE TUPLE DATA BLOCK NOT COUNTING THE WORDS FOR THE HEADER TUPLE DATA BLOCK. THIS MAY EXCEED THE VALUE IMPLIED BY NELT TO ALLOW GROWTH SPACE, IN WHICH CASE THE EXTRA WORDS CONTAIN THE SPECIFIER FOR THE APPROPRIATE NIL VALUE.

IF THE NELT VALUE IS CORRECTLY SET (AS INDICATED BY THE IS←NELTOK SETTING), THEN THE VALUE CORRESPONDING TO THE INDEX VALUE NELT-1 MUST NOT CONTAIN NIL (SO THAT THE NELT VALUE CORRESPONDS TO THE CARDINALITY OF A TUPLE).

4.2.2. SPECIAL TUPLES

SPECIAL TUPLES (AS OPPOSED TO STANDARD TUPLES) ARE REPRESENTED BY A SPECIFIER WHICH HAS:

TYPE: T←STUPLE
VALUE: POINTER TO APPROPRIATE TUPLE DATA BLOCK

THERE ARE THREE SUBTYPES OF SPECIAL TUPLES, THE HEDRTYPE VALUES OF THE CORRESPONDING DATA BLOCKS DISTINGUISH THE CASE AT HAND.

4.2.2.1. PACKED TUPLES

THE PACKED TUPLE DATA BLOCK CONSISTS OF A STANDARD FORMAT HEADER WHICH HAS (IN ADDITION TO THE STANDARD FIELDS):

HEDRTYPE: H←PTUPLE
PTBITS: NUMBER OF BITS PER VALUE
PTVALS: NUMBER OF VALUES PER WORD
PTVECT: POINTER TO VALUE VECTOR (SEE NEXT SECTION)

THE VALUES ARE STORED FROM RIGHT TO LEFT IN EACH WORD, FITTING AS MANY VALUES AS POSSIBLE IN EACH WORD. THUS THE SETTING OF PTVALS CAN BE COMPUTED FROM THE PTBITS SETTING AND THE WORD LENGTH. IT IS STORED TO AVOID THE TIME FOR THIS CALCULATION. IF THERE ARE UNUSED BITS, THEY ARE SET TO ZEROES.

4.2.2.1.1. PACKED VALUES

PACKED VALUES APPEARING IN PACKED TUPLES (AND ELSEWHERE) ARE IN ONE OF TWO FORMATS AS INDICATED BY THE MAGNITUDE OF THE VALUE STORED (WHICH IS ALWAYS INTERPRETED AS A POSITIVE INTEGER).

IF THE STORED VALUE IS LESS THAN OR EQUAL TO THE SYSTEM CONSTANT PACKMAX, THEN IT IS A ZERO-ORIGIN INDEX INTO THE VECTOR OF VALUES REFERENCED BY PTVECT. IN THE CASE WHERE THE PACKED VALUES ARE ELEMENTS OF A CONSTANT SET, THIS VECTOR CONTAINS ELEMENTS OF THE SET. FOR SMALL INTEGER VALUES, THE VECTOR CONTAINS SUCCESSIVE INTEGER VALUES.

A PACKED VALUE OF 0 IS USED TO INDICATE THE NIL VALUE. THE CORRESPONDING ZEROTH ELEMENT OF THE CONSTANT SET VECTOR CONTAINS A NIL VALUE WITH APPROPRIATE FORM.

IF THE STORED VALUE IS GREATER THAN PACKMAX, THEN IT CORRESPONDS TO A SMALL INTEGER VALUE WHICH IS EQUAL TO ONE LESS THAN THE STORED VALUE (E.G. INTEGER VALUE 512 IS STORED AS 513 IN THE PACKED FIELD).

4.2.2.2. REAL TUPLE

A REAL TUPLE CONTAINS UNTYPED REAL VALUES (SEE SECTION 3.). A REAL TUPLE DATA BLOCK HAS:

HEDRTYPE: H←RTUPLE

A REAL TUPLE DATA BLOCK IS IDENTICAL IN FORMAT TO A STANDARD TUPLE EXCEPT THAT THE VALUES STORED ARE UNTYPED REALS (OR THE NIL UNTYPED VALUE) RATHER THAN STANDARD TYPED VALUES.

4.2.2.3. INTEGER TUPLE

AN INTEGER TUPL CONTAINS UNTYPED INTEGER VALUES (SEE SECTION 3.). AN INTEGER TUPLE DATA BLOCK HAS:

HEDRTYPE: H←ITUPLE

AN INTEGER TUPLE DATA BLOCK IS IDENTICAL IN FORMAT TO A STANDARD TUPLE EXCEPT THAT THE VALUES STORED ARE UNTYPED INTEGERS (OR THE NIL UNTYPED INTEGER) RATHER THAN STANDARD TYPED VALUES.

4.2.3. NULL TUPLE

UNLIKE THE PREVIOUS SYSTEM, THE NULL TUPLE DOES NOT HAVE A SPECIAL VALUE. IT IS SIMPLY A TUPLE WHOSE DATA BLOCK HEADER WORD HAS A ZERO NELT VALUE WITH IS←NELTOK SET ON.

4.2.4. PAIR

A TUPLE OF TWO ELEMENTS IS CALLED A PAIR. IT IS STORED IN THE SAME WAY AS AN ORDINARY TUPLE WITH THE NELT FIELD CONTAINING A VALUE OF 2 TO INDICATE THAT TWO ELEMENTS ARE PRESENT (I.E. ELEMENTS WITH SUBSCRIPTS 1 AND 2). SMALL TUPLES IN GENERAL, AND PAIRS IN PARTICULAR, USUALLY DO NOT HAVE GROWTH SPACE ALLOCATED. THUS THE VALUE IN ALLOC IS USUALLY THE SAME AS THE VALUE IN NELT.

4.3. SET AND MAP FORMATS

THERE ARE SEVERAL FORMATS FOR SETS AND MAPS CORRESPONDING TO POSSIBLE REPR DECLARATIONS. IN THIS DESCRIPTION, THE USE OF THE WORD SET IS RESERVED FOR SETS OTHER THAN MAPS (ALTHOUGH THE SETL TYPE FOR ALL THESE OBJECTS IS SET).

IN ADDITION TO THE TYPING OF OBJECTS AS SETS OR MAPS BY THE APPEARENCE OF RELEVANT REPR DECLARATIONS, OBJECTS ARE DYNAMICALLY CONVERTED BETWEEN THESE FORMATS WHERE APPROPRIATE. FOR EXAMPLE, A

SET OF PAIRS IS CONVERTED TO MAP FORMAT IF IT IS USED AS A MAP, AND THE INVERSE CONVERSION OCCURS IF A NON-PAIR ELEMENT IS ADDED TO AN OBJECT STORED IN MAP FORMAT (SINCE MAPS CAN ONLY CONTAIN PAIRS). THIS DYNAMIC CONVERSION IS TRANSPARENT TO THE SETL PROGRAM, SINCE A SET OF PAIRS AND THE CORRESPONDING MAP ARE SEMANTICALLY EQUIVALENT.

SET AND MAP DATA BLOCKS HAVE SEVERAL STANDARD FIELDS (IN ADDITION TO THE FIELDS WHICH ARE COMMON TO ALL NON-PRIMITIVE DATA):

IS←MAP:	ON IF TYPE IS MAP RATHER THAN SET
IS←BASED:	ON FOR BASED MAPS AND SETS
IS←MMAP:	ON FOR MAPS WHICH HAVE IS←MULTI ON EVERYWHERE
IS←SMAP:	ON FOR MAPS WHICH HAVE IS←MULTI OFF EVERYWHERE
IS←ELSET:	ON FOR SET OF ELEMENTS, OR MAP FROM ELEMENTS
HASHTB:	POINTER TO HASH TABLE HEADER BLOCK

THE HASHTB FIELD POINTS TO THE HASH TABLE FOR THE SET OR MAP IF IT IS UNBASED, AND TO THE HASH TABLE OF THE BASE IF IT IS BASED.

IS←ELSET DISTINGUISHES THE SPECIAL CASES OF A SET WHOSE MEMBERS ARE ELEMENTS OF SOME BASE, OR MAPS WHOSE DOMAIN IS ELEMENTS OF SOME BASE. THESE OBJECTS CAN BE TREATED IN A SPECIALLY EFFICIENT WAY IN SOME SITUATIONS.

4.3.1. HASH TABLE STRUCTURE

SETS AND MAPS IN VARIOUS FORMATS ARE REPRESENTED BY HASH TABLES. THE BASIC FORMAT OF ALL HASH TABLES IS CONSISTENT THROUGHOUT THE DATA STRUCTURES, ALTHOUGH THE MANNER IN WHICH INFORMATION IS STORED IN THE INDIVIDUAL ELEMENT BLOCKS OF THE HASH TABLE DEPENDS ON THE USAGE.

THE HASH TABLE POINTER (HASHTB) IN THE SET HEADER POINTS TO A HASH TABLE HEADER DATA BLOCK WHICH CONTAINS THE FOLLOWING FIELDS:

HEDRTYPE:	H←HT
CLINK:	NOT USED (0)
LOGNHEDR:	LOG(BASE 2) OF NUMBER OF HASH HEADERS
NEB:	NUMBER OF ELEMENT BLOCKS IN HASH TABLE

THE NUMBER OF HASH HEADERS IS ALWAYS A POWER OF 2, WITH A MINIMUM VALUE OF 1 (CORRESPONDING TO A LOGNHEDR VALUE OF 0) AND A MAXIMUM VALUE OF MAXHEDRS (CORRESPONDING TO A LOGNHEDR VALUE OF LMAXHEDRS). MAXHEDRS AND LMAXHEDRS ARE SYSTEM CONSTANTS.

THE NEB FIELD CONTAINS THE NUMBER OF ELEMENT BLOCKS IN THE HASH TABLE, EXCLUDING THE TEMPLATE BLOCK, AND EXCLUDING ANY UNUSED HASH HEADERS. IN THE CASE OF UNBASED MAPS, THIS VALUE MAY NOT BE THE SAME AS THE NELT VALUE IN THE SET HEADER.

THE HASH TABLE PROPER CONSISTS OF A COLLECTION OF BLOCKS CALLED ELEMENT BLOCKS (EB). THERE IS ONE EB FOR EACH ENTRY IN THE HASH TABLE, AND A SPECIAL DUMMY EB, CALLED THE HASH TABLE TEMPLATE

BLOCK. THE TEMPLATE BLOCK IMMEDIATELY FOLLOWS THE HASH TABLE HEADER DATA BLOCK IN MEMORY.

THE NUMBER OF WORDS IN EACH EB VARIES WITH THE CONTEXT IN WHICH THE HASH TABLE IS USED.

THE FOLLOWING FIELDS OCCUR IN EVERY ELEMENT BLOCK:

HEDRTYPE:	TYPE OF HASH TABLE EB
GLINK:	(=EBSIZE) NUMBER OF WORDS IN EB
EBSPEC:	SPECIFIER FOR ELEMENT
EBLINK:	LINK POINTER USED TO CHAIN EBS OF A HASH TABLE
IS←EBBASE:	SET ON ONLY FOR BASE SET HASH TABLE EBS
IS←EBHEDR:	SET ON FOR EBS WHICH ARE HASH HEADERS
IS←EBTEMP:	SET ON ONLY FOR THE TEMPLATE, OFF ELSEWHERE
IS←EBFREE:	SET ON ONLY FOR UNUSED HASH HEADER

THE EBVALUE FIELD IS A COMPOSITE FIELD WHICH INCLUDES ALL FIELDS FOUND IN A NORMAL VALUE SPECIFIER. IT MAY BE THE CASE THAT EBVALUE CORRESPONDS TO AN ENTIRE SETL WORD, BUT THIS IS NOT REQUIRED.

THE EBLINK FIELD IS USED TO CHAIN THE EBS OF A HASH TABLE TOGETHER IN ONE LONG LINKED LIST, AS FURTHER DESCRIBED BELOW.

THE IS←EBBASE FLAG ALLOWS DETERMINATION OF WHETHER VALUES IN ELEMENT FORMAT (5.) ARE ELEMENTS OF A BASE OR NOT.

THE IS←EBHEDR FLAG IDENTIFIES THE START OF EACH HASH CHAIN.

THE NUMBER OF SETL WORDS REQUIRED FOR THESE STANDARD FIELDS OF AN EB (WHICH IS THE MINIMUM POSSIBLE SIZE FOR AN EB WITH NO EXTRA FIELDS) IS GIVEN BY THE SYSTEM CONSTANT EBHNW.

THE TEMPLATE EB OCCURS IMMEDIATELY FOLLOWING THE HASH TABLE HEADER WORD AND IS A DUMMY EB WITH ITS FIELDS SET AS FOLLOWS:

EBVALUE:	APPROPRIATE NIL VALUE
EBLINK:	POINTS TO FIRST HASH HEADER
IS←EBBASE:	SET ON ONLY FOR BASE SET TEMPLATE
IS←EBHEDR:	SET ON (A SPECIAL CASE!)
IS←EBTEMP:	SET ON
IS←EBFREE:	SET ON

THE HASH TABLE PROPER CONSISTS OF A CONTIGUOUS SEQUENCE OF ELEMENT BLOCKS CALLED HASH HEADERS. THE NUMBER OF HASH HEADERS CAN BE OBTAINED FROM THE LOGNHEDR FIELD OF THE HASH TABLE HEADER WORD. THE EBLINK FIELD OF THE TEMPLATE POINTS TO THE FIRST HASH HEADER EB. OFTEN THE TEMPLATE BLOCK WILL IMMEDIATELY PROCEED THE HASH HEADERS, BUT THIS IS NOT REQUIRED. HASH HEADER EBS ALWAYS HAVE THE IS←HEDR FLAG ON.

EACH HASH HEADER IS CHAINED (USING THE EBLINK FIELD) TO A LIST OF EBS. THE CHAIN OF EBS FROM ONE HASH HEADER CORRESPOND TO THOSE ELEMENTS WHICH HASH TO THE GIVEN HEADER POSITION.

THE EBS OF THE HASH CHAIN ARE CHAINED USING THE EBLINK FIELD OF THE FIRST WORD OF THE EB. THE END OF EACH CHAIN LINKS BACK TO THE NEXT HASH HEADER EB. THE LAST BLOCK ON THE LAST CHAIN CONTAINS AN EBLINK VALUE WHICH POINTS BACK TO THE TEMPLATE BLOCK. THUS THE EBS OF A SET ARE IN ONE LONG CIRCULAR LIST. THE END OF THE LIST IS DETECTED BY TESTING THE IS←EBTEMP FLAG WHICH IS ONLY ON FOR THE TEMPLATE BLOCK ITSELF.

IF A HASH HEADER IS UNUSED (I.E. NO VALUES HASH TO A PARTICULAR HEADER CHAIN), THEN ITS EBVALUE FIELD CONTAINS THE NIL VALUE, THE IS←EBHEDR BIT IS ON, AND THE EBLINK FIELD POINTS TO THE NEXT HASH HEADER (OR POINTS TO THE TEMPLATE EB IN THE CASE OF THE LAST HASH HEADER). THIS IS THE ONLY TIME THAT NIL VALUES CAN OCCUR IN EBVALUE. ALL NIL VALUES IN EBVALUE HAVE FORMS AND TYPES APPROPRIATE TO THE CONTEXT.

EVERY HASH TABLE HAS AT LEAST ONE HASH HEADER. IN THE CASE OF A NULL SET, THERE WILL BE ONE HASH HEADER WHICH IS UNUSED AND CONTAINS THE NIL VALUE AS USUAL.

THIS FORMAT RESULTS IN THE ELEMENTS OF THE HASH TABLE BEING CHAINED TOGETHER IN ONE LONG LIST WHILE RETAINING THE POSSIBILITY OF HASHED ACCESS FOR A SEARCH. NOTE THAT THE END OF A GIVEN HASH CHAIN IS DETECTED BY ENCOUNTERING AN EB WHOSE FIRST WORD HAS IS←HASHHEDR ON.

4.3.2. SET FORMATS

ALL SET FORMATS ARE REPRESENTED BY A SPECIFIER WHICH HAS:

TYPE:	T←SET
VALUE:	POINTER TO SET DATA BLOCK

THE IS←MAP BIT OF THE REFERENCED DATA BLOCK IS ALWAYS OFF. THE HEDRTYPE OF THE DATA BLOCK INDICATES THE PARTICULAR FORMAT OF SET REFERENCED.

4.3.2.1. UNBASED SET

THIS SECTION DESCRIBES THE FORMAT OF UNBASED SETS NOTE THAT THIS INCLUDES SETS OF ELEMENTS OF A BASE IN THE CASE WHERE THE KEYWORD BASED DID NOT APPEAR IN THE REPR. SUCH OBJECTS ARE STORED IN STANDARD UNBASED FORMAT WITH THE APPROPRIATE SPECIFIERS IN ELEMENT OF BASE FORMAT.

THE SET HEADER DATA BLOCK HAS:

HEDRTYPE:	H←USET

THE HASHTB FIELD OF THE DATA BLOCK (WHICH CONTAINS NO FIELDS OTHER THAN THE STANDARD ONES) POINTS TO A STANDARD FORMAT HASH TABLE (4.3.1.).

THE EB HAS:

HEDRTYPE: H←EBS

THE EBSPEC FIELDS OF THESE ELEMENT BLOCKS CONTAIN THE VALUES OF SUCCESSIVE ELEMENTS OF THE SET IN STANDARD SPECIFIER FORMAT.

THE EBSPEC FIELD OF THE TEMPLATE CONTAINS THE NIL VALUE TYPED AND FORMED CONSISTENTLY WITH THE TYPE OF ELEMENT IN THE SET.

4.3.2.2. INTEGER SET

AN INTEGER SET DATA BLOCK HAS:

HEDRTYPE: H←UISET

THE HASHTB FIELD POINTS TO A HASH TABLE HEADER BLOCK. THE EBS IN THIS HASH TABLE HAVE:

HEDRTYPE: H←EBIS
EBIVAL: UNTYPED INTEGER VALUE

NOTE THAT THESE EBS DO NOT HAVE AN EBSPEC FIELD SINCE THE VALUE APPEARS IN THE EBIVAL FIELD.

4.3.2.3. REAL SET

A REAL SET DATA BLOCK HAS:

HEDRTYPE: H←URSET

THE HASHTB FIELD POINTS TO A HASH TABLE HEADER BLOCK. THE EBS IN THIS HASH TABLE HAVE:

HEDRTYPE: H←EBRS
EBRVAL: UNTYPED REAL VALUE

NOTE THAT THESE EBS DO NOT HAVE AN EBSPEC FIELD SINCE THE VALUE APPEARS IN THE EBRVAL FIELD.

4.3.2.4. BASE SET

BASE SETS ARE REPRESENTED BY A SPECIFIER IN THE STANDARD FORMAT (THESE SPECIFIERS APPEAR IN BASE ARRAYS). THE DATA BLOCK FOR A BASE SET CONTAINS TWO SPECIAL FIELDS IN ADDITION TO THE STANDARD FIELDS:

HERTYPE: H←BASE
BLINK: USED BY GARBAGE COLLECTOR TO LINK BASES
RLINK: USED BY GARBAGE COLLECTOR TO LINK REMOTE OBJECTS

NLMAPS: NUMBER OF LOCAL (UNPACKED) MAPS ON BASE

THE LENGTH OF THIS EXPANDED BLOCK IS GIVEN BY THE SYSTEM CONSTANT BASEHNV.

THE HASHTB FIELD OF THE BASE SET DATA BLOCK, AS WELL AS THE HASHTB FIELD OF ALL OBJECTS BASED ON IT POINT DIRECTLY TO THE BASE SET HASH HEADER DATA BLOCK.

THE ELEMENT BLOCKS IN A BASE SET HASH TABLE CONTAIN THE FOLLOWING FIELDS:

HEDRTYPE: H←EBB
EBSPEC: VALUE OF BASE SET ELEMENT
EBLINK: EB LINK POINTER
IS←EBBASE: ALWAYS ON FOR BASE SET EBS
IS←EBHEDR: ON IN HASH HEADER EBS
IS←EBTEMP: ON FOR TEMPLATE BLOCK ONLY
IS←EBFREE: ON FOR UNUSED HASH HEADERS
EBINDEX: INDEX VALUE FOR REMOTE BASED OBJECTS
EBBASE: POINTER BACK TO BASE SET HEADER
EBHASH: HASH VALUE FOR ELEMENT
EBBSET: BIT STRING FOR BASED SETS

THE EBINDEX VALUES RANGE FROM 1 UP AND ARE ALLOCATED CONSECUTIVELY AS NEW EBS ARE ADDED TO THE BASE SET. THESE INDEX VALUES ARE USED TO REFERENCE THE REQUIRED ELEMENT OF REMOTE OBJECTS BY INDEXING.

THE EBBASE FIELD IS A BACKREFERENCE WHICH ENSURES THAT THE BASE FOR OBJECTS IN ELEMENT OF BASE FORMAT (SEE 5.) CAN EASILY BE OBTAINED.

THE EBHASH VALUE IS ALWAYS SET CORRECTLY, EXCEPT FOR UNUSED HASH HEADERS.

EBBSET IS A BIT FIELD WHICH USES UP THE REMAINING BITS (ZERO OR MORE) LEFT OVER IN THE INDEX WORD. IT IS USED TO CONTAIN AS MANY LOCAL BASED SET MEMBERSHIP BITS AS POSSIBLE TO AVOID ALLOCATION OF EXTRA EB WORDS FOR THIS PURPOSE UNNECESSARILY. THE NUMBER OF BITS IN THIS FIELD IS GIVEN BY THE SYSTEM CONSTANY EBBSETSIZE.

THE NUMBER OF WORDS REQUIRED FOR THIS EXPANDED EB FORMAT IS GIVEN BY THE SYSTEM CONSTANT EBBNV.

IF THERE ARE LOCAL OBJECTS ASSOCIATED WITH THE BASE, THEN ADDITIONAL WORDS MAY BE ALLOCATED FOLLOWING THE STANDARD FIELDS. THESE WORDS ARE USED TO HOLD THE VALUES OF LOCALLY BASED OBJECTS AS DESCRIBED IN THE INDIVIDUAL SECTIONS ON LOCALLY BASED OBJECTS. EACH UNIQUE LOCAL OBJECT IS ASSIGNED (STATICALLY) A GIVEN WORD OR BIT FIELD IN THE BASE SET EB TO CONTAIN THE VALUE. SOME OF THESE WORDS CONTAIN SPECIFIERS AND MUST BE PROCESSED BY THE GARBAGE COLLECTOR, OTHERS CONTAIN BIT STRINGS OR UNTYPED DATA WHICH MUST BE IGNORED. THE ENTRIES FOR STANDARD LOCAL MAPS CONTAIN POINTERS WHICH MUST BE RELOCATED BY THE GARBAGE COLLECTOR. ALL SUCH

ENTRIES OCCUR IMMEDIATELY AFTER THE INITIAL WORDS. THE NLMAPS OF THE BASE HEADER DATA BLOCK GIVES THE NUMBER OF SUCH ENTRIES IN EACH EB.

THE FIELDS OF THE TEMPLATE BLOCK ARE SET AS FOLLOWS:

HEDRTYPE:	H←EBB
EBVALUE:	APPROPRIATE NIL VALUE
EBLINK:	POINTS TO FIRST HASH HEADER
IS←EBBASE:	ON
IS←EBHEDR:	ON
IS←EBTEMP:	ON
IS←EBFREE:	ON
EBINDEX:	NEXT INDEX VALUE TO BE ASSIGNED
EBBASE:	POINTS TO THE TEMPLATE BLOCK AS USUAL
EBHASH:	CONTAINS ZEROES (HASH IS MEANINGLESS)
EBBSET:	CONTAINS ZERO BITS

NOTE THAT IF THE INDEX OF THE TEMPLATE BLOCK IS USED TO ACCESS A REMOTE OBJECT, THE INDEX WILL ALWAYS BE OUT OF RANGE AND A NIL VALUE WILL BE OBTAINED AS REQUIRED.

REMAINING WORDS IN THE TEMPLATE BLOCK (CORRESPONDING TO LOCAL OBJECT VALUES) ARE SET IN A MANNER APPROPRIATE TO THE PARTICULAR LOCAL OBJECT TYPE AS DESCRIBED IN THE INDIVIDUAL SECTIONS ON LOCAL MAPS.

NOTE THAT THE INDEX VALUE 0 IS NEVER USED. THE CORRESPONDING ZEROTH ENTRY IN REMOTE OBJECTS IS THE TEMPLATE FOR THE REMOTE OBJECT AND IS SET IN THE SAME MANNER AS THE HASH TABLE TEMPLATE BLOCK FIELD IN THE BASE FOR A CORRESPONDING LOCAL OBJECT.

4.3.2.5. PLEX BASES

A SPECIAL CASE OF BASE SETS IS THE PLEX BASE, WHICH CONSISTS OF LONG ATOM VALUES ONLY. THE LONG ATOM DATA BLOCKS FOR THESE VALUES CONTAIN ADDITIONAL FIELDS CORRESPONDING TO THE VALUES OF LOCAL MAPS DEFINED ON THE BASE, AS IN THE FORMAT OF AN ELEMENT BLOCK, THE LOCAL MAPS REPRESENTED BY STANDARD SPECIFIERS (I.E. THOSE WHICH MUST BE PROCESSED BY THE GARBAGE COLLECTOR) MUST COME FIRST. THE FIELD LA←NLMAPS GIVES THE NUMBER OF SUCH SPECIFIERS IN EACH LONG ATOM VALUE.

4.3.2.6. REMOTE SET

A REMOTE SET DATA BLOCK CONTAINS A STANDARD FORM SET HEADER WITH THE FOLLOWING FIELDS:

HEDRTYPE:	H←RSET
RS←MAXI:	MAXIMUM INDEX VALUE

THIS HEADER IS IMMEDIATELY FOLLOWED BY A BIT STRING WHICH

REPPESENTS THE VALUE OF THE REMOTE SET. THE LENGTH OF THIS BIT STRING (AS GIVEN BY RS←MAXI) MUST BE AT LEAST AS GREAT AS THE MAXIMUM INDEX FOR ANY VALUE IN THE BASE SET WHICH IS CONTAINED IN THE REMOTE SET (PLUS ONE SINCE INDEXES START AT ZERO). THE KTH BIT INDICATES THE MEMBERSHIP IN THE REMOTE SET OF THE BASE ELEMENT WHOSE INDE IS K. THE BIT IS ON IF THE ELEMENT IS IN THE REMOTE SET AND OFF IF IT IS NOT. THE FIRST BIT IN THE STRING (CORRESPONDING TO INDEX VALUE 0) IS NEVER USED, AND IS ALWAYS SET TO 0.

THE BITS ARE ARRANGED FROM RIGHT TO LEFT IN SUCCESSIVE WORDS. THE LAST (PARTIALLY FILLED) WORD CONTAINS UNUSED HIGH ORDER BITS.

THE SYSTEM CONSTANT RS←BPW GIVES THE NUMBER OF BITS STORED IN EACH WORD.

4.3.2.7. LOCAL SET

A LOCAL SET DATA BLOCK CONTAINS A STANDARD FORM SET HEADER WITH THE FOLLOWING FIELDS:

HEDRTYPE: H←LSET
LS←WORD: OFFSET TO WORD IN EB CONTAINING VALUE
LS←BIT: POSITION OF MEMBERSHIP BIT IN EB WORD

LS←BIT IS A BIT POSITION FROM THE LOW ORDER END OF THE WORD (LEAST SIGNIFICANT BIT NUMBERED 1).

THE SINGLE BIT IN THE INDICATED POSITION OF THE BASE EB INDICATES WHETHER THE BASE ELEMENT IS IN THE LOCAL SET OR NOT. IT IS ON IF THE ELEMENT IS IN THE LOCAL SET AND OFF IF NOT. NOTE THAT THERE IS ROOM FOR ONE OR MORE LOCAL SET BITS IN THE INDEX WORD OF THE BASE (IN THE EBBSET FIELD).

THE CORRESPONDING BIT IN THE TEMPLATE BLOCK OF THE BASE SET HASH TABLE IS ALWAYS OFF.

4.3.2.8. CONSTANT SETS

CONSTANT SETS ARE STORED IN EXACTLY THE SAME FORMAT AS NORMAL SETS EXCEPT THAT THEY ARE NEVER MODIFIED (AND CANNOT BE CONVERTED TO MAP FORMAT). FOR EACH CONSTANT SET, AN INDEX VECTOR IS BUILT FOR USE BY PACKED INDEX VALUES (SEE SECTION 4.2.2.1.1.). THE ZEROETH ELEMENT OF THIS VECTOR CONTAINS THE NIL VALUE. SUCCESSIVE ELEMENTS CONTAIN THE VALUES OF THE SET ELEMENTS IN SET ELEMENT FORMAT.

THERE IS NO LINK BETWEEN THIS VECTOR AND THE HASH TABLE SINCE NONE IS EVER NEEDED.

IF THE CONSTANT SET IS USED AS A BASE, THEN ITS HASH TABLE CONTAINS INDEX VALUES AS USUAL. THESE INDEX VALUES MATCH THE

INDEX VALUES USED IN THE INDEX VECTOR. THIS MEANS THAT PACKED VALUES IMMEDIATELY YIELD THE BASE INDEX WITHOUT REFERENCE TO THE INDEX VECTOR.

4.3.3. MAP FORMATS

ALL MAPS ARE REPRESENTED BY A SPECIFIER WHICH HAS:

TYPE: T←MAP
VALUE: POINTER TO MAP DATA BLOCK

THE MAP DATA BLOCK HEDRTYPE DETERMINES THE PARTICULAR MAP FORMAT.

4.3.3.1. MAP IMAGE REPRESENTATION

ALTHOUGH MAPS ARE SIMPLY SETS OF PAIRS IN SETL SEMANTICS, THE INTERNAL STORAGE FORM IS SUBSTANTIALLY DIFFERENT. IN PARTICULAR, IN THE CASE OF A MULTI-VALUED MAP, THE SET OF PAIRS WITH A COMMON HEAD IS GROUPED TOGETHER.

THE IMAGE OF A MAP FOR A PARTICULAR DOMAIN VALUE IS THUS EITHER A SINGLE VALUE OR A SET OF VALUES. THE LATTER CASE MUST BE DISTINGUISHED FROM A SINGLE VALUE WHICH HAPPENS TO BE A SET. THIS DISTINCTION IS THE FUNCION OF THE IS←MULTI BIT WHICH APPEARS IN ALL SPECIFIERS (BUT IS ONLY MEANINGFUL IN THIS CONTEXT).

IF THE IS←MULTI BIT IS OFF IN THE SPECIFIER REPRESENTING THE IMAGE VALUE, THEN THE MAP IS SINGLE VALUED FOR THE CORRESPONDING DOMAIN VALUE, AND THE SPECIFIER VALUE IS THE RANGE VALUE.

IF THE IS←MULTI BIT IS ON IN THE SPECIFIER REPRESENTING THE IMAGE VALUE, THEN THE SPECIFIER REPRESENTS THE VALUE OF A SET WHOSE MEMBERS ARE THE RANGE VALUES CORRESPONDING THE DOMAIN VALUE.

IF THE MAP IS SINGLE VALUED, THEN TWO REPRESENTATIONS ARE POSSIBLE. EITHER THE SINGLE RANGE VALUE WITH IS←MULTI OFF OR A SINGLETON SET WITH IS←MULTI ON.

THE IS←SMAP BIT OF THE MAP HEADER BLOCK IS SET ON IF ALL IMAGE VALUES HAVE IS←MULTI OFF. THIS ALSO IMPLIES THAT THE MAP IS SINGLE VALUED.

THE IS←MMAP BIT OF THE MAP HEADER BLOCK IS SET ON IF ALL IMAGE VALUES HAVE IS←MULTI ON. THIS DOES NOT IMPLY THAT THE MAP IS MULTI-VALUED EVERYWHERE SINCE SOME OR ALL OF THE RANGE SETS MAY BE SINGLETONS.

IF BOTH IS←SMAP AND IS←MMAP ARE OFF, THEN EITHER REPRESENTATION COULD BE USED AT SINGLE VALUED POINTS. HOWEVER, THE LIBRARY STANDARDIZES IN THIS CASE TO SET IS←MULTI ONLY IN THE CASE OF MULTI-VALUED POINTS.

4.3.3.2. UNBASED MAP

THE UNBASED MAP DATA BLOCK CONTAINS ONLY STANDARD FIELDS AND HAS:

HEDRTYPE: H←UMAP

THE HASHTB FIELD POINTS TO A HASH TABLE WHICH CONTAINS THE MAP VALUES.

THE ELEMENT BLOCKS IN THIS HASH TABLE HAVE:

HEDRTYPE: H←EBM
EBIMAG: MAP IMAGE VALUE

THERE IS ONE EB FOR EACH UNIQUE DOMAIN ELEMENT. ITS EBSPEC FIELD CONTAINS THE SPECIFIER FOR THE DOMAIN VALUE. THE EBIMAG FIELD CONTAINS THE IMAGE VALUE WITH THE IS←MULTI BIT INDICATING THE FORMAT AS DESCRIBED IN 4.3.3.1.

4.3.3.3. UNBASED INTEGER MAP

AN UNBASED INTEGER MAP DATA BLOCK HAS A STANDARD SET HEADER WITH

HEDRTYPE: H←UIMAP

THE ELEMENT BLOCKS OF THE HASH TABLE HAVE:

HEDRTYPE: H←EBIM
EBIIM: RANGE VALUE (UNTYPED INTEGER)

THE DOMAIN VALUE IS IN EBSPEC AS USUAL, THE IMAGE VALUE IS AN UNTYPED INTEGER VALUE STORED IN THE EBIIM FIELD. ALL INTEGER MAPS ARE SINGLE VALUED (MUTLI-VALUED REAL MAPS WOULD HAVE TO BE STORED IN STANDARD FORMAT).

4.3.3.4. UNBASED REAL MAP

AN UNBASED REAL MAP DATA BLOCK HAS A STANDARD SET HEADER WITH:

HEDRTYPE: H←URMAP

THE ELEMENT BLOCKS OF THE HASH TABLE HAVE:

HEDRTYPE: H←EBRM
EBRIM: RANGE VALUE (UNTYPED REAL)

THE DOMAIN VALUE IS IN EBSPEC AS USUAL, THE IMAGE VALUE IS AN NTYPED REAL VALUE STORED IN THE EBRIM FIELD. ALL REAL MAPS ARE SINGLE VALUED (MULTI-VALUED REAL MAPS WOULD HAVE TO BE STORED IN STANDARD FORMAT).

4.3.3.5. REMOTE MAP

A REMOTE MAP DATA BLOCK CONSISTS OF A STANDARD FORMAT SET HEADER WHICH HAS:

HEDRTYPE: H←RMAP

THE HASHTB FIELD OF THIS HEADER POINTS TO THE HASH TABLE HEADER DATA BLOCK FOR THE CORRESPONDING BASE SET.

THE HEADER IS IMMEDIATELY FOLLOWED BY A STANDARD FORMAT (T←TUPLE) TUPLE, COMPLETE WITH TUPLE HEADER BLOCK.

THE KTH ELEMENT OF THIS TUPLE CONTAINS THE MAP IMAGE VALUE, WITH IS←MULTI SHOWING THE FORMAT AS DESCRIBED IN 4.3.3.1.

THE VALUE IN MAXINDX (I.E. THE TUPLE LENGTH) MUST BE AT LEAST AS LARGE AS THE LARGEST INDEX VALUE FOR WHICH THE MAP IS DEFINED ON THE CORRESPONDING ELEMENT.

NOTE THAT THE TUPLE WHICH IS PART OF THE MAP DATA BLOCK DOES NOT CONTAIN A BASE ARRAY. THE BASE ARRAY FOR THE MAP IS ALWAYS ASSOCIATED WITH THE MAP BLOCK ITSELF.

4.3.3.6. REMOTE PACKED MAP

A REMOTE PACKED MAP DATA BLOCK HAS:

HEDRTYPE: H←RPMAP

IT IS IMMEDIATELY FOLLOWED BY A TUPLE OF VALUES AS FOR AN UNPACKED REMOTE MAP, EXCEPT THAT THE TUPLE IS IN PACKED TUPLE FORM (4.2.2.1.)

4.3.3.7. REMOTE REAL MAP

A REMOTE REAL MAP DATA BLOCK HAS:

HEDRTYPE: H←RRMAP

IT IS IMMEDIATELY FOLLOWED BY A TUPLE OF VALUES AS FOR AN UNPACKED REMOTE MAP, EXCEPT THAT THE TUPLE IS IN REAL TUPLE FORM (4.2.2.2.)

4.3.3.8. REMOTE INTEGER MAP

A REMOTE INTEGER MAP DATA BLOCK HAS:

HEDRTYPE: H←RIMAP

IT IS IMMEDIATELY FOLLOWED BY A TUPLE OF VALUES AS FOR AN UNPACKED REMOTE MAP, EXCEPT THAT THE TUPLE IS IN REAL TUPLE FORM (4.2.2.3.)

4.3.3.9. LOCAL MAP

A LOCAL MAP DATA BLOCK HAS A STANDARD FORMAT SET HEADER WITH ADDITIONAL FIELDS:

```
HEDRTYPE:   H←LMAP
LS←WORD:    OFFSET TO WORD IN EB CONTAINING VALUE
```

THE LENGTH OF THIS EXPANDED HEADER IS GIVEN BY THE SYSTEM CONSTANT HL←LPMAP.

LS←WORD SHOWS THE LOCATION OF THE WORD IN EACH EB OF THE BASE WHICH CONTAINS THE VALUE OF THE MAP. THE REFERENCED WORD CONTAINS THE MAP IMAGE WITH IS←MULTI INDICATING THE FORMAT AS USUAL (4.3.3.1.)

4.3.3.10. LOCAL PACKED MAP

A PACKED LOCAL MAP DATA BLOCK CONSISTS OF A STANDARD FORMAT SET HEADER WITH ADDITIONAL FIELDS: THE SECOND WORD OF THE HEADER CONTAINS THE FOLLOWING ADDITIONAL FIELDS, DESCRIBING THE PACKING:

```
HEDRTYPE:   H←LPMAP
LS←WORD:    OFFSET TO WORD IN EB CONTAINING VALUE
LS←BIT:     STARTING BIT NUMBER FOR FIELD
LS←BITS:    NUMBER OF BITS IN FIELD
LS←VECT:    POINTER TO VALUE VECTOR
```

THE LENGTH OF THIS EXPANDED HEADER BLOCK IS GIVEN BY THE SYSTEM CONSTANT LPMAPNW.

THE LS←BIT VALUE IS THE BIT NUMBER OF THE LOW ORDER BIT OF THE FIELD (LS BIT NUMBERED 1).

THE SIGNIFICANCE OF THE REFERENCED BIT STRING VALUE AND ITS RELATION TO THE LPMVECT FIELD IS THE SAME AS FOR THE VALUES IN A PACKED TUPLE (SEE SECTION 4.2.2.1.1.).

THE CORRESPONDING FIELD IN THE TEMPLATE BLOCK OF THE BASE SET HASH TABLE CONTAINS ALL ZERO BITS (THE PACKED REPRESENTATION OF THE NIL VALUE).

4.3.3.11. LOCAL REAL MAP

A REAL LOCAL MAP BLOCK IS IN STANDARD FORMAT WITH ADDITIONAL FIELDS:

HEDRTYPE: H←LRMAP
LS←WORD: OFFSET TO WORD IN EB CONTAINING VALUE

THE WORD IN THE BASE SET EBS CONTAINS EITHER THE APPROPRIATE REAL VALUE, OR THE NIL REAL VALUE.

THE CORRESPONDING WORD IN THE TEMPLATE BLOCK OF THE BASE SET HASH TABLE CONTAINS THE NIL UNTYPED REAL VALUE.

4.3.3.12. LOCAL INTEGER MAP

AN INTEGER LOCAL MAP BLOCK IS IN STANDARD FORMAT HEADER WORD WITH ADDITIONAL FIELDS:

HEDRTYPE: H←LIMAP
LS←WORD: OFFSET TO WORD IN EB CONTAINING VALUE

THE WORD IN THE BASE SET EBS CONTAINS EITHER THE APPROPRIATE INTEGER VALUE, OR THE NIL INTEGER VALUE.

THE CORRESPONDING WORD IN THE TEMPLATE BLOCK OF THE BASE SET HASH TABLE CONTAINS THE NIL UNTYPED INTEGER VALUE.

5. SET ELEMENT FORMAT

THERE IS A SPECIAL FORMAT FOR A REFERENCE TO AN ELEMENT OF A HASH TABLE. THERE ARE THREE CASES:

1) REFERENCE TO EB OF A BASE SET
2) REFERENCE TO EB OF AN UNBASED SET
3) REFERENCE TO EB OF AN UNBASED MAP

VALUES OF THIS TYPE ARE REPRESENTED BY A SPECIFIER WHICH HAS:

TYPE: T←ELMT
VALUE: POINTER TO CORRESPONDING EB

THE VALUE FIELD MAY BE USED OBTAIN INFORMATION FROM THE EB INCLUDING:

VALUE OF SET OR BASE ELEMENT, OR MAP DOMAIN ELEMENT
FORWARD POINTER (EBLINK) FOR ITERATION
VALUE OF UNBASED MAP FOR GIVEN DOMAIN ELEMENT
INDEX WORD FROM BASE FOR REMOTE REFERENCE
VALUE FROM BASE FOR LOCAL REFERENCE

IF AN OBJECT IN T←ELMT FORMAT OCCURS IN A CONTEXT (E.G. ADDITION) WHICH REQUIRES AN ACTUAL VALUE, THEN A DEREFERENCING OPERATION IS PERFORMED TO OBTAIN THE CORRESPONDING SET ELEMENT OR MAP DOMAIN ELEMENT. NOTE THAT IF THE SET OR MAP CONTAINS VALUES IN ELEMENT FORMAT, THIS DEREFERENCING MAY NEED TO BE DONE REPEATEDLY TO OBTAIN THE ACTUAL VALUE.

VALUES IN ELEMENT OF FORMAT OCCUR IN THE FOLLOWING CONTEXTS:

1) WHEREVER THE REPR ELEMENT OF BASE IS USED
2) FOR ITERATIONS THROUGH MAPS AND SETS

THEY MIGHT ALSO BE USED MORE GENERALLY IF REPR DECLARATIONS WERE ALLOWED TO SPECIFY ELEMENT OF NON BASE SET OR ELEMENT OF DOMAIN OF UNBASED MAP.

NOTE THAT THE CASE OF ELEMENT OF FORMAT REFERENCING A BASE IS DISTINGUISHED BY THE SETTING OF THE IS←BASE FLAG IN HASH TABLE ELEMENT BLOCKS.

PLEX BASES ARE A SPECIAL CASE IN THAT AN ITEM WHICH IS AN ELEMENT OF A PLEX BASE IS REPRESENTED SIMPLY AS THE LONG ATOM VALUE OF THE BASE. SUCH A VALUE MAY BE USED INTERCHANGABLY AS THE ATOM VALUE IT REPRESENTS, AND AS A POINTER TO OBTAIN LOCAL MAP VALUES.

6. ITERATOR FORMATS

EN ITERATING THROUGH A MAP OR BASE, A STANDARD VALUE SPECIFIER IS USED TO CONTROL THE ITERATION. IN THE CASES OF SETS AND MAPS (BUT NOT TUPLES), IT IS POSSIBLE TO USE THIS SPECIFIER TO OBTAIN THE CURRENT VALUE, AS WELL AS TO OBTAIN THE NEXT VALUE OF THE ITERATION. HOWEVER, THE SETL RUNTIME SYSTEM USUALLY MAINTAINS TVO SEPARATE ITERATOR VALUES EXCEPT IN CERTAIN CASES VHERE THIS OPTIMIZATION IS POSSIBLE AND DESIRABLE.

6.1. UNBASED SET ITERATOR

ITERATORS FOR UNBASED SETS ARE IN THE FOLLOVING FORMAT:

TYPE:	T←ELMT
VALUE:	POINTER TO CORRESPONDING ELEMENT BLOCK

THE INITIALIZATION FOR THE ITERATOR CONSISTS OF SETTING THE VALUE FIELD TO POINT TO THE TEMPLATE BLOCK. THE ITERATION IS THEN PERFORMED BY USING THE VALUE FIELD TO LOCATE THE THE EB CONTAINING THE POINTER TO THE NEXT EB.

IF THE ACTUAL VALUE IS REQUIRED IT CAN BE OBTAINED FROM THE REFERENCED ELEMENT BLOCK, AND IT VILL OFTEN BE ADVANTAGEOUS TO OBTAIN THIS VALUE ONCE ON EACH ITERATION AND STORE IT IN A SEPARATE LOCATION.

NOTE THAT THIS FORMAT CORRESPONDINGS EXACTLY TO THE ELEMENT OF SET FORMAT PREVIOUSLY DESCRIBED.

6.2. BASE ITERATOR

THE FORMAT OF A BASE SET ITERATOR IS SIMILAR TO AN UNBASED SET ITERATOR. THE RESULTING ELEMENT OF FORMAT OBJECT CAN BE USED TO OBTAIN ANY REQUIRED INFORMATION FROM THE BASE SET ELEMENT BLOCK.

THE ITERATOR IS INITIALIZED BY POINTING TO THE TEMPLATE BLOCK, AND ITERATED IN THE SAME MANNER AS FOR A NORMAL SET.

6.3. UNBASED MAP ITERATOR

THE FORMAT OF AN UNBASED MAP ITERATOR IS:

TYPE:	INDICATES TUPLE
VALUE:	POINTER TO PAIR VALUE

ELEMENT 1 OF THE PAIR IS IN ELEMENT OF MAP DOMAIN FORMAT:

TYPE:	T←ELMT

VALUE: POINTER TO EB IN MAP

ELEMENT 2 OF THE PAIR IS IN ONE OF TWO FORMATS:

TYPE: TYPE OF RANGE ELEMENT F(X)
VALUE: VALUE OF RANGE ELEMENT F(X)

THIS CASE IS FLAGGED BY IS-RANGE BEING OFF IN THE TUPLE DATA BLOCK FOR THE PAIR.

OR

ELEMENT FROM SET F(/X/) IN APPROPRIATE ITERATOR FORMAT. IN THIS CASE, THE IS-RANGE FLAG OF THE TUPLE HEADER BLOCK FOR THE PAIR IS SET ON.

NOTE THAT THE VALUE OF THE FIRST ELEMENT OF THE PAIR IS IN STANDARD ELEMENT OF DOMAIN OF MAP FORMAT.

6.4. REMOTE MAP ITERATOR

A REMOTE MAP ITERATOR IS IN THE SAME FORMAT AS AN ORDINARY MAP ITERATOR EXCEPT THAT THE FIRST WORD OF THE PAIR IS IN BASE ITERATOR FORMAT AND REFERENCES THE EB OF THE BASE SET.

6.5. LOCAL MAP ITERATOR

A LOCAL MAP ITERATOR IS IN THE SAME FORMAT AS AN ORDINARY MAP ITERATOR EXCEPT THAT THE FIRST WORD OF THE PAIR IS IN BASE ITERATOR FORMAT AND REFERENCES THE EB OF THE BASE SET.

6.6. REMOTE SET ITERATOR

AN ITERATOR FOR A REMOTE SET IS IN THE SAME FORMAT AS A BASE SET ITERATOR FOR THE CORRESPONDING BASE SET.

6.7. LOCAL SET ITERATOR

AN ITERATOR FOR A LOCAL SET IS IN THE SAME FORMAT AS A BASE SET ITERATOR FOR THE CORRESPONDING BASE SET.

6.8. TUPLE ITERATOR

TUPLE ITERATOR IS SIMPLY AN INDEX VALUE STORED AS A SHORT INTEGER.

6.9. DOMAIN ITERATORS

IF A PROGRAM SPECIFIES DIRECTLY OR INDIRECTLY AN ITERATION THROUGH THE DOMAIN OF A MAP OR TUPLE, THEN A SPECIAL DOMAIN ITERATOR FORMAT IS USED WHICH REFERS TO THE MAP OR TUPLE ITSELF (RATHER THAN ACTUALLY CREATING THE DOMAIN AS A SET AND ITERATING THROUGH IT).

6.10. UNBASED MAP DOMAIN ITERATOR

THE DOMAIN ITERATOR IS IN STANDARD ELEMENT OF MAP DOMAIN FORMAT:

TYPE:	T←ELMT
VALUE:	POINTER TO CORRESPONDING ELEMENT BLOCK

6.11. BASED MAP DOMAIN ITERATOR

THIS HAS THE SAME FORMAT AS A BASE SET ITERATOR FOR THE CORRESPONDING BASE.

6.12. TUPLE DOMAIN ITERATOR

THE DOMAIN OF A TUPLE IS SIMPLY THE SET OF INTEGERS FROM 1 TO THE NUMBER OF ELEMENTS IN THE TUPLE. IF AN ITERATION THROUGH THE DOMAIN OF A TUPLE IS PERFORMED, THE CORRESPONDING DOMAIN ITERATOR IS SIMPLY A STANDARD FORMAT SHORT INTEGER (2.1.1.).

7. FORM TABLE

ALL OBJECTS IN THE SYSTEM HAVE AN ASSOCIATED FORM, WHICH IS THE REPR MINUS ANY INDICATION OF THE SPECIFIC BASE SETS INVOLVED IN BASING RELATIONS (BUT INCLUDING FULL KNOWLEDGE OF THE OCCURENCE AND TYPE OF BASING).

THE FORM TABLE IS A SEPARATE VECTOR (FORMTAB) WHICH CONTAINS ENTRIES REPRESENTING THESE FORM VALUES.

FOR PRIMITIVE DATA, THE CORRESPONDING FORM TABLE ENTRY IS DETERMINED BY THE CONTEXT IN WHICH THE VALUE APPEARS (THERE IS NO DIRECT LINK FROM PRIMITIVE VALUES TO THEIR CORRESPONDING FORM TABLE ENTRIES).

FOR NON-PRIMITIVE DATA (TUPLES, SETS, MAPS), THE DATA BLOCK FOR THE HEADER CONTAINS A FORM FIELD WHICH IS AN INDEX TO THE RELEVANT FORM TABLE ENTRY.

THE WORDS OF THE FORM TABLE, WHICH ARE NOT NECESSARILY THE SAME LENGTH AS OTHER SETL WORDS, CONTAIN THE FOLLOWING FIELDS:

FT←TYPE:	TYPE CODE F←XXX INDICATING FORM TYPE
FT←ELMT:	FORMTAB INDEX FOR ELEMENT TYPE
FT←DOM:	FORMTAB INDEX FOR MAP DOMAIN TYPE
FT←IM:	FORMTAB INDEX FOR MAP IMAGE TYPE
FT←RSET:	FORMTAB FOR MAP RANGE SET TYPE
FT←NBASEA:	NUMBER OF ENTRIES IN BASE ARRAY
FT←IS←NIL:	ON IF VALUE MAY BE NIL
FT←IS←LIM:	ON IF LIMITING INFORMATION PRESENT
FT←LOV:	LOW BOUND OF LIMIT INFORMATION
FT-HIGH:	HIGH BOUND OF LIMIT INFORMATION
FT←MAPC:	INDICATES SMAP,MAP,MMAP SETTING
FT←HASHOK:	ON IF IS←HASHOK MAY BE SET
FT←NELTOK:	ON IF IS←NELTOK MAY BE SET
FT←ESHARE:	ON IF ELEMENT CAN SHARE BASE ARRAY
FT←DSHARE:	ON IF DOMAIN CAN SHARE BASE ARRAY
FT←ISHARE:	ON IF IMAGE CAN SHARE BASE ARRAY

FT←TYPE CONTAINS A CODE INDICATING THE ENTRY TYPE. THE FOLLOWING SECTIONS LIST THE CODES ACTUALLY USED.

FT←ELMT IS USED FOR SETS, MAPS AND TUPLES. FOR SETS AND TUPLES, IT IS THE FORM OF THE CORRESPONDING ELEMENT. FOR MAPS, IT IS THE FORM OF THE TUPLE USED TO REPRESENT CORPESPONDING PAIRS. FOR MIXED TUPLES, IT POINTS TO A SPECIAL AUXILIARY TABLE CALLED MTTAB.

FT←DOM IS USED FOR MAPS TO BIVE THE FORM OF THE DOMAIN TYPE.

FT←IM IS USED FOR MAPS TO GIVE THE FORM OF THE IMAGE. FOR SMAPS AND8MAPS, THIS IS THE FORM OF SINGLE RANGE ELEMENTS. FOR MMAPS, T IS THE FORM FOR THE CORRESPONDING SET OF RANGE ELEMENTS.

FT←RSET IS USED FOR MAPS TO GIVE THE FORM FOR THE SET OF RANGE

VALUE ELEMENTS (WHICH WOULD APPEAR IF IS←MULTI WERE ON IN AN IMAGE VALUE). FOR MMAPS, FT←RSET IS THE SAME AS FT←IM.

NOTE: THE VALUES OF FT←DOM, FT←IM AND FT←RSET ARE ALSO USED FOR SETS WHICH COULD POSSIBLY BE CONVERTED TO MAPS AND REPRESENT THE SETTINGS FOR THIS CONVERTED OBJECT. IN OTHER SETS (AND ALL OTHER TYPES OF OBJECTS), THESE FIELDS ARE SET TO ZEROES.

FOR UNBASED SETS: THE FORM OF THE CORRESPONDING MAP IF THE SET WERE TO BE CONVERTED TO MAP.

FOR ALL SMAPS: THE FORM OF THE CORRESPONDING MAP

FT←NBASEA CONTAINS THE NUMBER OF ENTRIES IN THE CORRESPONDING BASE ARRAY (4.1.). THE BASE ARRAYS THEMSELVES CANNOT BE LOCATED FROM THE FORM TABLE, SINCE TWO OBJECTS WHICH HAVE DIFFERENT BASINGS, BUT ARE OTHERWISE IDENTICAL, HAVE THE SAME FORM TABLE ENTRY. FOR ALL PRIMITIVE OBJECTS, AND SETS AND TUPLES WITH NO BASED COMPONENTS, FT←NBASEA IS ZERO.

FT←IS←LIM IS SET TO INDICATE THAT FT←LOW AND FT←HIGH CONTAIN INFORMATION WHOSE SIGNIFICANCE VARIES WITH THE ENTRY TYPE AS FOLLOWS:

FOR AN INTEGER: FT←LOW = MINIMUM VALUE
FT←HIGH = MAXIMUM VALUE

FOR STRING: FT←LOW = MINIMUM LENGTH OF STRING
FT←HIGH = MAXIMUM LENGTH

FOR TUPLES: FT←LOW = MINIMUM LENGTH OF TUPLE
FT←HIGH = MAXIMUM LENGTH

FT←MAPC USED FOR MAPS ONLY TO INDICATE DECLARED TYPE:

FT←SMAP INDICATES SMAP
FT←MMAP INDICATES MMAP
FT←MAP INDICATES MAP

FT←HASHOK AND FT←NELTOK INDICATE THAT THE CORRESPONDING BITS IN THE HEADER MAY BE SET. IF THE FT BIT IS OFF, THEN THE CORRESPONDING HEADER BIT IS ALWAYS OFF (AND NEED NEVER BE CHECKED).

FT←ESHARE, FT←DSHARE, FT←ISHARE ARE SET TO INDICATE THAT THE CORRESPONDING SUBOBJECTS CAN SHARE THE BASE ARRAY OF THE CONTAINING OBJECT.

7.1. FORM TYPE CODES

SHRT INT	F←INT
SHORT STRING	F←STRING
SHORT ATOM	F←ATOM
SUBR	F←SUBR

FUNC	F←FUNC
LONG ATOM	F←LATOM
ELEMENT	F←ELMT
LONG INT	F←LINT
LONG STRING	F←LSTRING
REAL	F←REAL
TUPLE	F←TUPLE
PACKED TUPLE	F←TUPLE
INTEGER TUPLE	F←ITUPLE
REAL TUPLE	F←RTUPLE
MIXED TUPLE	F←MTUPLE
UNBASED SET	F←USET
INTEGER SET	F←ISET
REAL SET	F←RSET
UNBASED MAP	F←UMAP
UNBASED INTEGER MAP	F←IMAP
UNBASED REAL MAP	F←RMAP
LOCAL SET	F←LSET
REMOTE SET	F←RSET
LOCAL MAP	F←LMAP
REMOTE MAP	F←RMAP
LOCAL PACKED MAP	F←LPMAP
LOCAL INTEGER MAP	F←LIMAP
LOCAL REAL MAP	F←LRMAP
REMOTE PACKED MAP	F←RPMAP
REMOTE INTEGER MAP	F←RIMAP
REMOTE REAL MAP	F←RRMAP
BASE SET	F←BASE
UNTYPED INTEGER	F←UINT
UNTYPED REAL	F←UREAL
GENERAL	F←GEN

7.2. MIXED TUPLE REFERENCE TABLE

MIXED TUPLES (TUPLES WHOSE ELEMENTS ARE FIXED, NON-IDENTICAL TYPES) USE AN AUXILIARY TABLE CALLED (MTTAB) TO INDICATE THE ELEMENT TYPES. THIS TABLE CONTAINS A SERIES OF ENTRIES WITH THE FOLLOWING FIELDS:

MT←FORM: FORM TABLE INDEX FOR ELEMENT
MT←OFFS: BASE ARRAY OFFSET FOR ELEMENT

A MIXED TUPLE VALUE IN THE FORM TABLE CONTAINS A POINTER INTO MTTAB (I.E. AN INDEX) WHICH INDICATES THE START OF THE CONTIGUOUS ENTRIES FOR THE MIXED TUPLE. ALL MIXED TUPLES ARE INDICATED TO BE LIMITED IN LENGTH, SO THE FT←LIM FIELD GIVES THE NUMBER OF ENTRIES.

THE MT←FORM VALUE IS THE INDEX OF THE FORM (IN FORMTAB) FOR THE CORRESPONDING ELEMENT TYPE.

THERASE ARRAY FOR A COMPONENT OF A MIXED TUPLE IS A CONTIGUOUS SUBSECTION OF THE BASE ARRAY FOR THE MIXED TUPLE ITSELF. THE LENGTH OF THIS SUBSECTION IS INDICATED BY THE FT←NBASEA FIELD OF

THE COMPONENT FORM, ITS STARTING LOCATION IS INDICATED BY THE OFFSET VALUE IN THE CORRESPONDING MT←OFFS FIELD. NOTE THAT MT←OFFS IS ALWAYS ZERO FOR THE ENTRY CORRESPONDING TO THE FIRST ENTRY OF THE MIXED TUPLE.

7.3. CONVERSION CONTROL TABLE

THE CONVERSION ROUTINE OF THE LIBRARY IS CONTROLLED BY A TABLE (CONVTAB) WHICH IS A VECTOR WHOSE ENTRIES CONTAIN THE FOLLOWING FIELDS:

CT←FORM1:	FORM OF INPUT TO CONVERSION
CT←FORM2:	FORM OF OUTPUT FROM CONVERSION
CT←FORMS:	ABOVE TWO FIELDS TOGETHER
CT←CASE:	CASE INDEX FOR CONVERSION ROUTINE
CT←IS←OK:	ON IF CONVERSION IS ALWAYS POSSIBLE

THE CONVERSION ROUTINE IS IN GENERAL FACED WITH THE TASK OF CONVERTING AN OBJECT FROM ONE FORM TO ANOTHER. THERE ARE MANY SPECIAL CASES WHICH REQUIRE SPECIAL SECTIONS OF CODE IN THE CONVERSION ROUTINE.

THE CONVERSION CONTROL TABLE MAPS INPUT AND OUTPUT FORMS INTO A CASE INDEX WHICH IS USED INSIDE THE CONVERT ROUTINE TO BRANCH TO THE APPROPRIATE SECTION OF CODE.

THE TABLE IS HASHED BASED ON THE CT←FORMS VALUE, USING SIMPLE LINEAR PROBING TO RESOLVE HASH CONFLICTS. IF AN ENTRY DOES NOT APPEAR IN THE TABLE, THE CORRESPONDING CONVERSION IS NOT POSSIBLE.