MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Lincoln Laboratory

20 July 1964

MEETING

TO: Distribution List

FROM: L. G. Roberts

SUBJ: Coral List Structure System; Introduction and Examples

SPEAKER: L. G. Roberts

TIME: 1:00 - 3:00, Thursday, July 23rd.

ROOM: A-258

Coral is a list structure language for the TX-2 which allows the user to create, associate, and delete variable length list structure blocks. The language is a set of M4 macro commands. In order to familiarize potential Coral users with the use of the macros a series of example problems will be examined.

This meeting will be the first of a series on Coral. The more sophisticated macros such as the class macros, will be discussed at a later time. However, concepts basic to their understanding will be discussed.

Larry Roberts

L. G. Roberts

LGR/jk

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Lincoln Laboratory

31 July 1964

TO: TX-2 Distribution List

FROM: L. G. Roberts

SUBJ: CORAL Meeting

This session for CORAL users will involve a short review, many examples and an introduction to the class concepts.

SPEAKER: L. G. Roberts

DATE: Monday, 3 August 1964

TIME: 1:00 - 3:00 p.m.

PLACE: A-258

jk

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Lincoln Laboratory

28 May 1964

TO: Users of Class Oriented Ring Associative Language (CORAL)

FROM: Larry, Alex

REF: Charts for Blockheads, NUBs, and Ring Macros

Charts for Blockheads and NUBs are attached and briefly explained below. A tabulation of available Ring Macros is also included and will be explained later.

Blockheads

There are now several types of blocks ($2^{17}-1$). Complete format details will be kept in a "Master Block" for each. The Blockhead gives fairly complete format information for some; less for others depending on the kind (type) of block. The type-numbering scheme includes all varieties: Blocks, Nubs, Quads etc. (Even Triads and Tubs if they are allowed to exist.)

When you ask the blockmaker macro for a block, you must specify the Type Number desired. The master block determines the number of ties. Where ever "ℓ" appears in the blockhead you can specify a data length that differs from the master spec. In all other cases, the master rules, "ℓℓ" is included in the Blockhead to tell where the data is without a master block reference.

Nubs etc.

The word "Nub" is now a class name. Where "nub" was used heretofore, we now use "plain nub" if such precision is required. There are 8 varieties of Nubs and two kinds of QUADS (Quad 4 and Quad 3). A Quad is a 4 register block with a Ring Start (Hen) on top (No Blockhead). A quad 4 is 4 tie registers. A quad 3 has 3 tie registers and a data register (which must come last). A quad may look like two nubs, but its real ancestor is the "BLOCK". Except for the top Hen, its tie registers have 2.9 = 0 and the data word of a Quad 3 is a full 36 bits.

Ring Macros

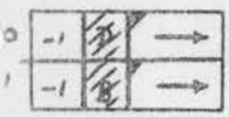
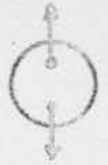
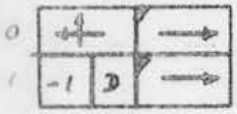
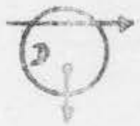
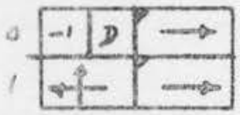
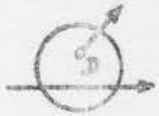
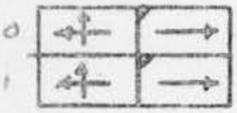
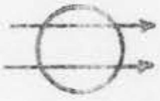
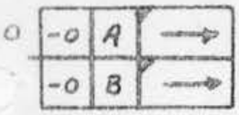
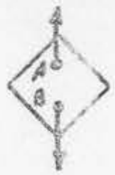
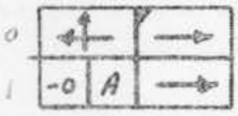
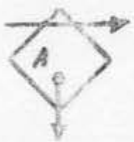
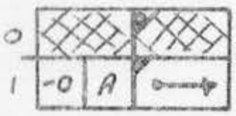
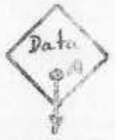
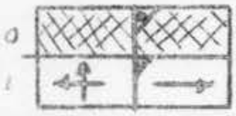
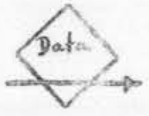
The attached list gives the "Macro" and "English" names for the current set of Standard Ring Macros. There is a growing amount of stability surrounding this list - people are using them. Detailed verbal descriptions of each ring macro will be prepared soon.

Alex

A. Vanderburgh

AV/41

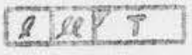
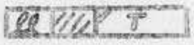
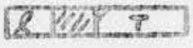
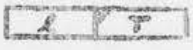
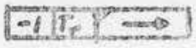
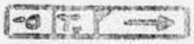
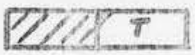
NUB CHART

Format	Name	Diagram	Number	Type Number Range
	Hub (See Note 9)		1000	360000 - 360777
	T stub		1000	361000 - 361777
	Y stub		1000	362000 - 362777
	Nub		1000	363000 - 363777
	Hub Item		1000	364000 - 364777
	T stub Item		1000	365000 - 365777
	STUD		1000	366000 - 366777
	DUB		1000	367000 - 367777

Notes:

1. Except for "full" data words in DUBS and STUDs, 2.9 must be 1.
2. If quarter 4 is -0 or -1, the tie register is a Ring Start (HEN).
3. "A,B" in HUB and T STIB ITEMS might be interpreted as the right half of a type number by some people someday.
4. " o—▸ " in diagrams is a Ring Start (HEN).
5. ——▸ (straight thru) is a (weep! sob!) chicken.
6. ▨ = Data
7. ✛ The left half of a chicken is either a back or hen tie.
8. STUDs and DUBs are now upside down - data is on top.
9. The two nine bit data words of a Hub are not available to ordinary Users.

Blockhead Chart

Kind of Block	Blockhead Format*	Block Use	Number of blocks	Range of type Nbrs. T
1.		Normal	200,000	0 - 177,777
2.		Fixed Data Length	100,000	200,000 - 277,777
3.		Variable Data "	40,000	300,000 - 337,777
4.		Length 256_d(page)	20,000	340,000 - 357,777
5.	NUBS		10,000	360,000 - 367,777
6.	Not Used yet		4,000	370,000 - 373,777
7. a.)		Quad 4	1,000	374,000 - 374,777
b.)		Quad 3	1,000	375,000 - 375,777
8.		Fixed. Last 100 reserved.	1,777	376,000 - 377,677
				376,700 - 377,776

*Notes:

1. ℓt = Number of Tie Registers +1

   ℓ = Total Lenght of Block.

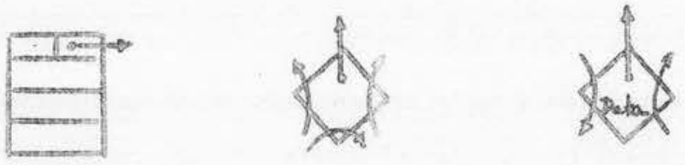
   T = Type Number

   T_r = Right half of type number.

   ▨ = Data

2. All Blocks, NUBS, and QUADS are located in memory starting at an Even numbered register.

3. Diagrams for Quads are generally one of the following:



General Form:	Name:	Most Common Case
MOVE		
P$\boxed{F}$→S	FLIP (NUB)	$\boxed{F}$
P$\boxed{D}$N→S	TO TOP OF DATA + N	$\boxed{D}$N
P$\boxed{M}$TP→S	FIND MASTER TYPE RING FOR THIS BLOCK OR TYPE	$\boxed{M}$TP
P$\boxed{>}$SB | ISB∧L→S	GO ROUND RING RIGHT	$\boxed{>}$SB
P$\boxed{<}$SB | ISB∧L→S	GO ROUND RING LEFT	$\boxed{<}$SB
P‖X | Y$\boxed{\supset}$SB→S	GO ROUND SUB CLASS OF P	$\boxed{\supset}$SB
P‖X | Y$\boxed{\in}$SB→S	GO ROUND SUPER CLASSES CONTAINING P	$\boxed{\in}$SB
P↑N→S	UP N	↑ N
P↓N→S	DOWN N	↓ N
P→N→S	RIGHT N	→ N
P←N→S	LEFT N	← N
P⤒N→S	TOP AND UP FROM BOTTOM N	⤒ N
P⤓N→S	TOP AND DOWN N	⤓ N
P⇥N→S	HEN AND RIGHT N	⇥ N
P⇤N→S	HEN AND LEFT N	⇤ N
TEST		
P$\boxed{H}$⊃YES | NO	HEN ?	$\boxed{H}$⊃YES
P$\boxed{=}$Q~¢⊃YES | NO	EQUAL ?	$\boxed{=}$Q⊃YES
P$\boxed{G}$Q~¢⊃YES | NO	GREATER ?	$\boxed{G}$Q⊃YES
P$\boxed{L}$Q~¢⊃YES | NO	LESS ?	$\boxed{L}$Q⊃YES
P$\boxed{?}$Q~¢>GR<LS=EQ	COMPARE	$\boxed{?}$Q>G<LS=EQ
P$\boxed{\approx}$⊃YES | NO	NUB ?	$\boxed{\approx}$⊃YES
P | N$\boxed{B}$BIT=J→K⊃YES | NO	BIT OF P+N=J? SET BIT TO K	$\boxed{B}$BIT=J⊃YES
P$\boxed{=}$TP⊃YES | NO	IS THIS TYPE TP ?	$\boxed{=}$TP⊃YES
DATA		
P$\boxed{K}$N~¢→S	CONTENTS	$\boxed{K}$
P$\boxed{T}$→S	TYPE	$\boxed{T}$
P$\boxed{I}$→S	BLOCK LENGTH	$\boxed{I}$
P$\boxed{J}$→S	LIST LENGTH	$\boxed{J}$

General Form:	Name.	Most Common use
CHANGE		
P⧁Q⇒S	PUT P RIGHT OF Q	⧁Q
P⧀Q⇒S	PUT P LEFT OF Q	⧀Q
P⊖Q | X⇒S	PUT P FIRST IN CLASS Q	⊖Q
P⊜Q | X⇒S	PUT P LAST IN CLASS Q	⊜Q
PⓉ⇒S	TAKE (HEN OR CHICKEN)	Ⓣ
P⊗⇒S	DELETE (BLOCK, HEN, CHICKEN)	⊗
PⒷTP∧L | TIE⇒S	<u>BLOCK MAKER</u> ALA MASTER AT P OR OF TYPE TP (L⇒ NON STANDARD LENGTH, TIE⇒TIE TO MASTER)	ⒷTP
TPⓂL×LL⇒S	DEFINE MASTER FOR TYPE TP	TPⓂL×LL
ⓃTP⇒S	MAKE NUB (TYPE TP)	Ⓝ TP
XΦL⇒S	START NEW LIST AT X, LENGTH L	XΦL
X⊝P | N≈q	STORE DATA X AT LIST LOCATION P+N	⊝ P
AE		
X$\underline{+}$Y≈q⇒Z	ADD X TO Y	$\underline{+}$ Y
X$\underline{-}$Y≈q⇒Z	SUBTRACT Y FROM X	$\underline{-}$ Y
X$\underline{\times}$Y∧N≈q⇒Z	MULTIPLY X TIMES Y, SCALE N	$\underline{\times}$ Y
X$\underline{/}$Y∧N≈q∩P⇒Z	DIVIDE X BY Y, SCALE N FIRST	$\underline{/}$ Y
X$\underline{\wedge}$Y≈q⇒Z	INTERSECT (MASK) X BY Y	$\underline{\wedge}$ Y
X$\underline{\vee}$Y≈q⇒Z	UNITE X WITH Y	$\underline{\vee}$ Y
X$\underline{\odot}$Y≈q⇒Z	DISTINGUISH (PARTIAL ADD) X AND Y	$\underline{\odot}$ Y
|X$\underline{\ \ }$|≈q⇒Z	ABSOLUTE VALUE OF X	|X$\underline{\ \ }$|
X$\underline{\Rightarrow}$Y⇒Z	LOAD STORE A WITH X IN Y AND Z	$\underline{\Rightarrow}$ Y
OTHER		
P⊠W1 | W2 | W3 | W4⇒S	GROUP DO W1, W2, W3, W4	⊠1 | W2
P◫Q	GO TO COMPUTED Q SAVE A, COMPUTE Q, RESTORE A, GO TO Q	◫Q
DO ‖ W ‖ N/α | α	DO W N/α TIMES WITH INDEX α	DO ‖ W ‖ N
SUBRTN | W	SUBROUTINE SAVE γ, DO W, RETURN BY γ	SUBRTN | W

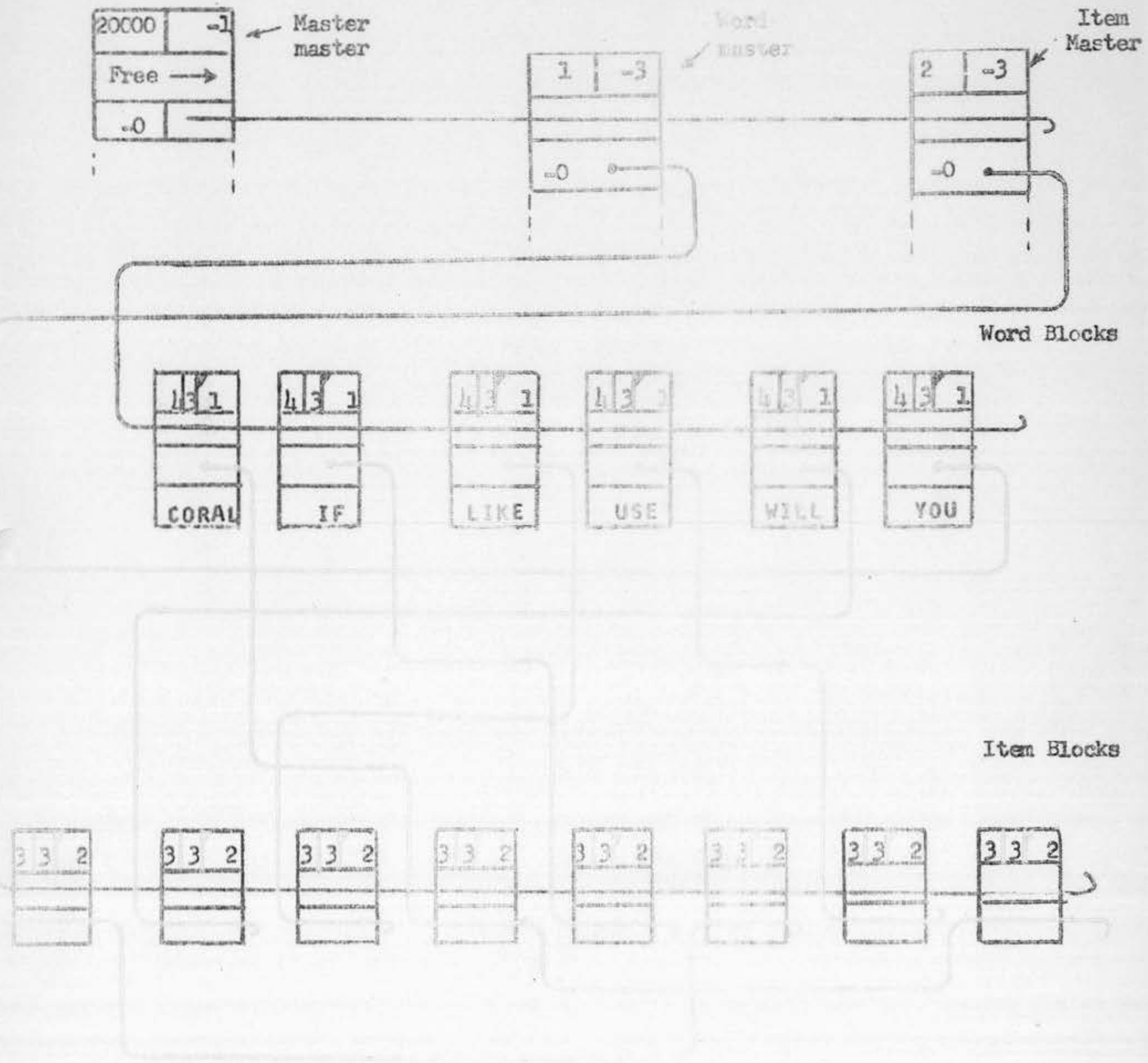
Here is a sample program in CORAL.
A Diagram of the rings it creates is
given on the next page. Good Luck.

Try to be of good cheer. A complete
description of each macro is in preparation.

```
⇒⇒DEF   TSD6|P
        LDB P
        CYB {6,}
        &TSD B
        STB P
        LDA {77}
        ITA B
⇒⇒EMD
⇒⇒DEF   SEQST|#
        RFD_#  #+1
        IOS_#  30010
⇒⇒EMD
ITEM=2
WORD=1
**EXAMPLE PROG TO TYPE TEXT IN AND OUT
**LIST OF 'WORDS' ORDERED IN SIMPLE DICTIONARY
**LIST OF 'ITEMS' (WORD USES) TIED TO WORD BLOCKS
SETUP→  0⊕20000                    **START LIST STRUCTURE AT REG 0
        ITEM⊛3×3                    **DEFINE ITEM BLOCK
        WORD⊛4×3→WDLST              **DEFINE WORD BLOCK AND SAVE TIE POINTER
**NOW COMPILE UP TO 6 TYPED CHARACTERS OR UNTIL SPACE, CR, OR STOP
TYPEIN→ SEQST|65                    **SET UP INPUT SEQ
        {-0}⇒INPUT                  **START INPUT WITH DELETE CODES
        DO‖((TSD6|INPUT)|⊟76⊃PRINT|(⊟70⊃DN|(⊟60⊃DN)))‖6      **ACCEPT INPUT
DN→     WDLST⊡((K2)⍰INPUT⊃NEW=HAVE)→WDP         **SEARCH FOR WORD
        ⇒WDP                        **WORD NOT FOUND, SAVE THEN POINTER
NEW→    (⊛WORD)⊛WDP→WDP             **MAKE WORD BLOCK AND TIE IN ORDER
        INPUT⊛WDP|2                 **DATA WORD STORED IN WORD BLOCK
HAVE→   ⊛ITEM|1                     **MAKE ITEM BLOCK AND TIE TO MASTER
        (⍖1)⊛(WDP⍖1)⇒TYPEIN         **TIE ITEM TO WORD BLOCK AND RETURN
**IF STOP CODE TYPE OUT TEXT
PRINT→  SEQST|66                    **SET UP OUTPUT SEQ
        (MITEM)⊡OUTSB⇒DICT          **FIND ITEM MASTER AND GO THROUGH ITEMS
OUTSB→  ((⍖1)⍏)K1→OUTPUT            **MOVE FROM ITEM TO WORD FOR DATA
OUTM→   DO‖((TSD6|OUTPUT)⊟77⊃OUTR)‖6   **TYPE OUT 6 DATA CHAR UNLESS DELETI
OUTR→   BPQ_λ 0                     **RETURN TO GO ROUND MACRO
**NEXT PRINT OUT DICTIONARY IN ORDER
DICT→   WDLST⊡(F&TSD {60}|(K2→OUTPUT)⇒OUTM)     **TYPE CR AND WORD
        RFD_65 SETUP
        JPD #
```

The diagram below is the Ring Structure Created by Larry's sample program if you type:

YOU WILL LIKE CORAL IF YOU USE CORAL



MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LINCOLN LABORATORY

TO: All Ranchhands and APEX Workers 17 July 1964

FROM: Larynx (Larry & Alx)

RE: CORAL — Class Oriented Ring Associative Language
(Formally known as Larry and Bert's RING MACROS)

FORWARD:

CORAL is a set of M4 Macro-instructions designed for the creation of data files as ring-structured tables and for data manipulation through flexible arrangement and variation of "ties" or "pointers." It is especially applicable to cases where there will be considerable categorization and recombination of data and where one set of data is strongly related to another.

For example:

1. Electronic Circuit Simulation — Graphical, Electrical, and Topological Parameters.
2. Programming Language Conversion — Flow Diagram, Symbolic Manuscript, Machine Language Code.

The document that follows is an attempt to introduce the jargon and use of CORAL. Neither this memo nor CORAL itself is complete as yet but in the interest of communication and coordination . . .

OUTLINE

DRAFT

I. Jargon

Block -- A group of registers

(Block Head)
("The Top")
-- The top register of a Block ("Top" is low numbered end, and is always on even numbered address.)

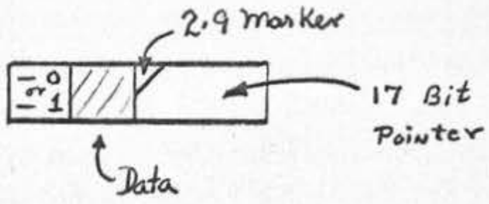
(Pointer)
(Tie)
-- A 17-bit word used to "Point" at something of interest. (i.e., it is the address or memory location of the thing it points to.)

List -- A set of blocks tied together with "Pointers" (i.e., Ties) that tell where the next block is.

Tie Register -- A register that contains the "Pointer" or "Tie."

Ring -- A list tied back to its beginning.

Ring Start
Tie Register
(Hen)
-- The first tie register of a Ring. It is also called a HEN. (RSTR = Rooster = Hen) Quarter 4 is either -0 or -1 for identification. See below:



A "Nothing" -- A "Nothing" is a register that cannot look like a Hen or Chicken. In pure form it contains -0 ,, 0.

Ring Element Tie Register (Chicken) -- Chicken (sorry -- no justification possible yet) A chicken is the ordinary tie register.

Tie Register Pair -- NUB -- One of a set of special 2-word blocks -- (See NUB chart)

Item -- A NUB that contains data of some sort. (See NUB chart)

List Structure -- Any set of blocks, rings, Nubs, etc. of this sort, i.e. tied up in rings & things.

Index or (Dictionary) -- A List Structure based on a string of character codes where one works through the structure by using each code in turn. i.e., the first letter determines the entry point, the second letter, how far around the first ring, the third, how far down the next ring, and so forth until the end. The data itself or a pointer to it would be found in the last block. (or NUB or ITEM as the case may be)

Class -- A group of things you wish to refer to by name. They will commonly be ordered automatically by the ring macros, and the structure will be designed for logarithnic search. Note all nubs of a given structure will have the same type number.

Free Ring
(or Free List) -- All available storage is tied together in as few blocks as possible on the "FREE" ring,

Pointer
Register -- A tagged register containing a pointer but outside any ring structure. Its pointer probably points AT some particular structure entry point. Its TAG might be thought of as the name for the whole structure.

Back Pointer -- A pointer in the left half of a CHICKEN TIE REGISTER that points to the closest previous (left) chicken that also has a Back Pointer. (See also HEN POINTER.)

Ring Start
Pointer
(Hen Pointer) -- A pointer in the left half of a CHICKEN TIE REGISTER that points to the HEN, (Ring START TIE REGISTER). Lefthand Pointers alternate (between BACK and HEN) in a "well formed" or "undisturbed" ring. This allows rapid access

to the RING START (HEN) without losing access to the previous ring element.

Master Block -- For each type number, the system will hold a MASTER BLOCK containing all block format information and perhaps other information if such is needed by the macros.

Type Number -- Each element of a given list structure - Blocks, NUBS, QUADS, etc. will have a Type Number. These numbers are to be assigned from the Blockhead Chart and with the co-operation of other users. A given block format and MASTER BLOCK can have but one Type Number (and vice versa). You can, of course, use as many blocks of the same type as you need.

MACRO
(Macro Instruction) -- An abbreviation for a flexible subprogram. The CORAL language is a coherent set of macros, designed for manipulation of RING STRUCTURES. (See LM 45, TX-2 Users Handbook, chapter 6, page 6-29).

Macro Definition)
Macro Name)
Dummy Parameter) See LM 45, TX-2 Users Handbook, page 6-29.

II. Terminators, Dummy Parameters, and Comments

A. Comments:

1.) Use parentheses to nest macros.

2.) Undefined parameters will be set to their value in the next nest, or the next, or the next, right back to the main program.

3.) The Accumulator is used for inter-macro communication. It usually contains a pointer (in the right half).

4.) Pointers will usually point to Tie Registers. Many macros count on this for correct operation. For example, if you are pointing at the Data portion of a block, the macros can not find the blockhead.

5.) Entrance to subroutine is usually just an address e.g. "SB". The macro creates JPQ SB. You can embellish it at your whim - e.g. h'SB_{χ} becomes h $BPQSB_{\chi}$, $^{2}SB_{\lambda}$ becomes $JES_{\lambda}SB$ etc. Use index λ for subroutine entrance, and BPQ_{λ} o (index λ) for return.

6.) The macros use index registers α, γ, and λ.

B. Terminators and Dummy Parameters that always mean the same thing:

1. → S — STA S --- Used to save the current pointer. S may have config. and/or index syllables. Check carefully before using a Macro.

2. ⌐ R — JPQ R --- Always at the end of the macro. R may have config. and/or index syllables. Check carefully before using a macro.

3. P (at left) LDA P --- Used as a way to get started. Not used if A already has the desired pointer. (One can often get started by use of a macro instead.)

4. $\sim q$ q is the desired configuration. If not specified, $q = 1$ in comparisons and $q = 0$ in arithmetic.

5. N N is usually a count. If an honest tag or expression containing such is given, the macro looks in that register for the count. If a macro is given, the generated number is used.

6. TP TP is usually a TYPE NUMBER although it may be the same as { Type Number}, or a Macro that computes the type number.

III. Individual Macros

MOVE Macros e.g. (Note: There is no protection, you can "move" right out of the block.)

P↑N→S←R

Macro NAME (or SYMBOL)	Common Use	English Name and Comment
↑	↑ N	"Move UP" N moves the current Pointer (in A) up the manuscript N Places (i.e. toward lower numbers). Changes A_{21}, C_{21}, D_{21} and Z_2.
↓	↓ N	"Move DOWN"
→	→ N	"MOVE RIGHT" N moves the pointer N elements around the ring to the right. This is the preferred direction. N can be as large as you wish, it just keeps going around. Changes A, γ , α.
←	← N	"MOVE LEFT" N. It takes longer to move to the left around a ring, for the Ties alternate and extra checking is required. Changes A, γ, α.
↧	↧ N	DOWN from the top N places. Starting point must be a Tie Register or Blockhead. There is no protection: You can move past the end of the block. If you use zero for N, you stay at the TOP. Changes A, C_{21}, D_{21}, α, γ , Z_2.

Macro NAME (or SYMBOL)	Common Use	English Name and Comment
$\overline{\uparrow}$	$\overline{\uparrow}$ N	Around the TOP and Up from the bottom N Places. Same comments as above apply. Note that if N is zero or missing, you stay at the top. Changes A, C_{21}, D_{21}, α, γ, Z_2.
$\overline{>}$	$\overline{>}$ N	Move N places to the right of the hen (Ring Start). If N is zero or missing, you will point to the hen. (Ring Start) Changes A, α, γ.
$\overline{<}$	$\overline{<}$ N	Move N places to the left of the Ring Start (HEN). (Same comments as above.) Changes A, B, α, γ.
$\boxed{F}$	$\boxed{F}$	FLIP to other side of NUB. Note that this amounts to $MKC_{1.1}$ A. Changes A_{21}.
$\boxed{D}$	$\boxed{D}$ N	Move to $N^{\underline{th}}$ Data register of the block. Be sure to move out of the data portion if you wish to use the macros that must have access to the TOP. Changes α, γ, A, B, C, D.
$\boxed{M}$	$\boxed{M}$ TP	Move to Master block for type number "TP". You will be pointing at the third register of the Master block. (i.e. at the start of the "Type Ring".)
	P $\boxed{M}$	If no type number is give, the macro will go to the TOP of the block to find one and then proceed to the master. If it can not find one, e.g. if Block is a nub or special block, then -1 is returned as an error signal.

Macro NAME (or SYMBOL)	Common Use	English Name and Comment
$\boxed{>}$	$\boxed{>}$ SB	Go Round Ring to the right, doing subroutine "SB" for each element. SB can also be a macro expression. Macro generates the jump to SB and sets up L for return via BPQ$_{\lambda}$o.
	$\boxed{>}$ SB ∧ L	L is the address or name of a push down List provided so that a recursive "go round" can be done conveniently. (See examples) Use JES $_{\gamma}$ to recurse.
$\boxed{<}$	$\boxed{<}$ SB	Go Round Left: Similar to Go Round Right except slower.

B. TEST MACROS

Notes:

1. Two arguments are compared. One comes into A via LDA P.
2. The other argument is specified by Q
   - a) If Q is a number, that number <u>is</u> the second argument.
   - b) If Q is a tag, the contents of Q is the second argument.
   - c) If Q is a macro, its result is the second argument.
3. The comparison is made under the aegis of configuration "q". For <u>this</u> <u>group</u> of macros, q = 1 if not specified.
4. YES (and/or NO) can be a macro, a Tag, or a Subroutine Entry. A macro is just evaluated as always. A Tag is used as an exit jump-no provision is made for return.

Macro Name	Use	Comment
$\boxed{=}$	P $\boxed{=}$Q⊃YES	Are they "EQUAL"? Note: They must be identical. -0 does not "equal" +0. Changes nothing!
$\boxed{G}$	P $\boxed{G}$Q⊃YES	Is $\mathrm{Arg}_P > \mathrm{Arg}_Q$? Is the argument from P algebraicly greater than that from Q? (Changes D and Z corresponding to q.)
$\boxed{L}$	P $\boxed{L}$Q⊃YES	Is $\mathrm{Arg}_P < \mathrm{Arg}_Q$? Is the argument from P algebraicly less than that from Q? (Changes D and Z corresponding to q.)
$\boxed{?}$	P $\boxed{?}$>GR<LS=EQ	Three exits, same rules as for YES and NO, (see note 4 above). GR for $\mathrm{Arg}_P > \mathrm{Arg}_Q$ LS for $\mathrm{Arg}_P < \mathrm{Arg}_Q$ and EQ for $\mathrm{Arg}_P = \mathrm{Arg}_Q$. But this time 0 = -0. (Changes D and Z corresponding to q.)
$\boxed{H}$	$\boxed{H}$⊃YES	Pointing at a HEN? (Ring Start Tie Register) (No config. "q", no Q) Changes A_{43} and α.
$\boxed{n}$	$\boxed{n}$⊃YES	Pointing at some kind of NUB? (No config. "q", no Q) Changes A_{43}, α, and γ.
$\boxed{B}$	P|N$\boxed{B}$BIT=J→K⊃YES|NO	BIT is the Bit number - in the form 2.7; P|N is where to find it, (N is a number up or down from P,) Question is, "Is BIT equal to J?" K is what the bit should be changed to. Hence, the macro reads: Bit of (P+N) = J ? ⊃ yes/No, set Bit to K Changes α.

$\boxed{\equiv}$	$\boxed{\equiv}TP\supset YES\mid NO$	Pointing at a block of type TP? (Naturally, you wouldn't be pointing at the Data part.) Changes A_{43}, B, C, D, α, γ and clears all overflow flops (Z_1, Z_2, Z_3, Z_4).

C. Data

Macro Name	Use	Comment
$\boxed{K}$	$\boxed{K}N\sim q$	This macro turns into a ^{q}LDA (a + N) where "a" is the right half of Acc and N is a number. Changes A via config. q, and α.
$\boxed{T}$	$P\boxed{T}\rightarrow S$	What Type of block is this? Changes A, B, C, D, α, γ and clears all overflow flip-flops.
$\boxed{I}$	$P\boxed{I}\rightarrow S$	What Length block is this?
$\boxed{J}$	$P\boxed{J}\rightarrow S$	What is the length of the Non Data part of this block?

Note: For the last 3 ($\boxed{T}$, $\boxed{I}$, and $\boxed{J}$) the following is true:

a) The answer to the question goes into the right half of Acc.

b) The original pointer goes in the right half of B and a pointer to the top of the block goes into the left half of B.

D. A. E. Macros

Name	Use	Comment
$\underline{+}$	$X\overline{\underline{+}}Y\sim q\to Z$	ADD
$\underline{-}$	$X\underline{-}Y\sim q\to Z$	SUB
$\underline{\times}$		
$\overline{\times}$	$X\underline{\times}Y\wedge N\sim q\to Z$	MUL, then SAB N places
$\underline{/}$	$X\underline{/}Y\wedge N\cap P\to Z$	SABN, then DIV, Qustient goes into Z. If overflow, go to P. Remainder is left in B and includes initial B contents not shifted out. If N is left out, B is cleared to $\underline{+}$ 0 depending on A.
$\underline{\wedge}$	$X\underline{\wedge}Y\sim q\to Z$	Interest (MASK, AND)
$\underline{\vee}$	$X\underline{\vee}Y\sim q\to Z$	Unite (inclusive OR)
$\underline{\circledvee}$	$X\underline{\circledvee}Y\sim q\to Z$	Distinguish (Partial Add, Exclusive OR)
$\underline{\perp}$	$\vert X\underline{\perp}\sim q\to Z$	Magnitude of x (Absolute Value, Make x Positive)
$\underline{\to}$	$X\underline{\to}Y\to Z$	Put contents of χ in Y and Z.

Note: 1. X, Y can be what you wish - Tag, Macro, configured, Indexed, etc.

2. N must not be a Macro.

3. q is the configuration used on the operation itself. (You can therefore use 3 configurations - on X , on Y, and on the operation.) Note that there are no restrictions on q - any config. is OK.

E. Misc. Macros

Name	Use	Comment
$\boxed{*}$	$P\boxed{*}W1 \mid W2 \mid W3 \mid W4 \to S$	Do W1, W2, W3, W4. The Macro definition merely lists the parameters as follows: W1 W2 W3 W4 They can therefore be whatever you wish. The macro itself changes nothing.
$\boxed{F}$	$P\boxed{F}Q$	"GO" to result of or contents of Q with contents of or result of P in right half of Acc. Changes A_{43}.
DO	$DO \mid W \mid N/Z \mid X$	Do "W" N/Z times using index X . Uses α if X is not given. Z is the index increment.

Note: 1. Any W will be whatever you write -- be it subroutine, macro, or what have you.

F. Change Macros!

Name	Use	Comment
⧁	P⧁Q→S	Put P to the right of Q. If Q is a nothing, it becomes a HEN.
⧀	P⧀Q→S	Put P to the left of Q.

Name	Use	Comment
ⓣ	Pⓣ	Take P. Take unties a chicken from its ring, reties the ring, and leaves Acc pointing at the taken chicken (now a nothing.) At the moment, take should not be used on HENS.
ⓜ	PⓜTP∧L | TIE→S	Make a block of type TP or of same type as block P points at. L is for non-standard Length. "TIE" is used if you wish a tie to the master in the second reg. of block. Acc. is left pointing at the 2nd. register of the block.
⊜	TP⊜L×LL→S	Block Type TP is defined to have L registers and (LL - 1) Tie Registers.
ⓝ	ⓝTP	Make nub of type TP.
⦶	X⦶L	Create Master Master block at X (Stand List Structure) of length L.
⊖	X⊖P | N~q	Store datum χ at (P + N) via config. q.

CLASS MACROS -- For Speed, Economy, Flexibility, and Convenience

1. Any Grouping of Data Blocks can be considered a "CLASS".

   a) The same set of subset of Blocks can be grouped several ways - hence forming several CLASSES.

   b) The order of the grouping may depend on the data itself, on the block type number, or on the order of insertion of the blocks.

   c) The grouping is done, of course, by ties -- the blocks exist but once.

2. A SUBGROUP is automatically generated when two or more elements have the same ordering criterium. The ordering within the subgroup will be pushdown. (Pushdown can be from either end.)

3. There are two macros for insertion of a data block into a CLASS grouping: "PUT FIRST", ↄ⃝ and 'PUT LAST", © . (They are analogous to Put Right ⧁ and Put left ⧀ .)

4. The "PUT IN CLASS" macros have parameters for specification of the ordering desired. This gives rise to 3 basic versions of the two macros: (illustrated for Put Last.)

a.) P©Q PUT P LAST IN Q

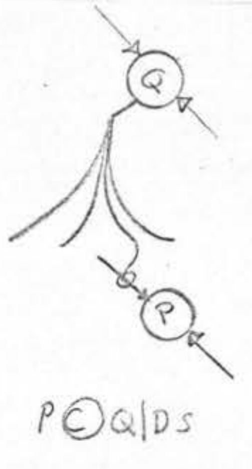
b.) P©Q|DS PUT P LAST IN Q ACCORDING TO DS

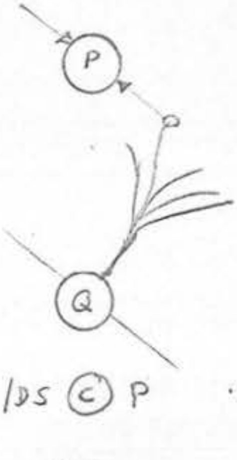
c.) Q|DS©P PUT P LAST IN Q ACCORDING TO DS

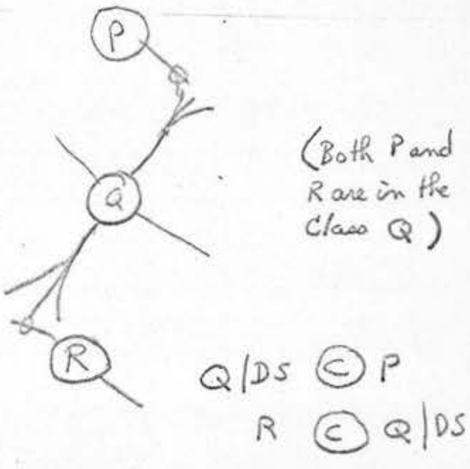
5. There is a difference between cases b and c above. Classes can be given Hierarchial Status. The class on the left of the symbol © is SUBORDINATE to the class on the right.

When DS is used, it is a macro or subroutine that conjures up a 36-bit ordering number - either from the block itself or by charging off to some other place. (Ordering numbers less than 36-bits must be sign extended.)

The logarithmic search structure can be diagrammed as a "TREE". If P©Q|DS is used, Q is superior, and the tree branches downward towards P. If Q|DS©P is used, P is superior, and the tree branches upward toward P. Note that a block can have two trees - one "up" and one "down" - with a common name.





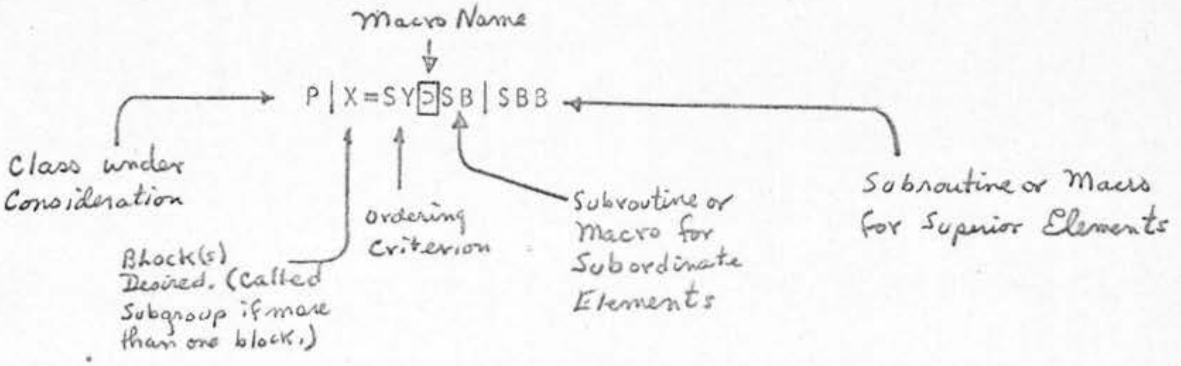


6. The hierarchial structure becomes significant when the Delete Macro " ⊗ " (circled Multiply Sign) is used. Delete in simple form deletes everything it can get at going downwards. Any block left "hanging" -- i.e. with nothing pointing to it is also deleted. Delete takes two forms:

P⊗	DELETE P
P⊗Q	DELETE P FROM Q

7. There are two "Go Around" Macros:

P$\boxed{\supset}$ GO AROUND P AND P$\boxed{\subset}$ GO ROUND CLASSES P BELONGS TO.


Macro Name
P | X=SY⊐SB | SBB
Class under Consideration
Block(s) Desired. (Called Subgroup if more than one block.)
Ordering Criterion
Subroutine or Macro for Subordinate Elements
Subroutine or Macro for Superior Elements

Although you can look at superior and subordinate elements separately, they are on the same ring and will be considered in the proper order regardless of their hierarchial status. (You can, of course, use P P$\boxed{\supset}$SB|SB if all elements are desired without distinction) The parameters are:

P - The class to be considered

X - A 36-bit number, the block desired. If there is more than one, they are considered in the order of generation (First to Last or Vice-Versa depending on Put Macro used) If no such element was found, or after the last one, the macro exigts to the next line on the manuscript and is pointing to the next higher element.

SY - SY is the ordering criterium. It is repeated here for speed. (It could have been remembered by Put, perhaps.)

SB - A subroutine or macro to be done for all subordinate blocks.

SBB - For all Superior blocks.

P$\boxed{\subset}$SB | SBB - "Go Around Classes P belongs to", SB is "done" for subordinate classes, SBB for superior classes (They could be the same subroutine.) The order is determined by the order of creation. (Each time P became a member of some class, that class went last on the ring of "classes P belongs to".)