IBM

OS/VS Multiprogramming Services

Student Materials

IBM

OS/VS Multiprogramming Services

Student Materials

Major Revision (September 1975)

PROGRAM ATTRIBUTES

TYPE	PROGRAM ACTION	SYSTEM ACTION
NOT REUSABLE	NONE.	ALWAYS LOADS NEW COPY.
SERIALLY REUSABLE	MUST RESTORE ANY WRITTEN-INTO AREAS TO THEIR ORIGINAL CONDITION.	WILL ALLOW MULTIPLE TASKS (WITHIN THE SAME REGION) TO USE THE SAME COPY. ONLY ONE TASK AT A TIME CAN HAVE ACCESS. (VS2 ONLY)
REENTERABLE	MODULE CANNOT BE WRITTEN INTO.	WILL ALLOW MULTIPLE TASKS TO USE THE SAME COPY CONCURRENTLY.

V.1.1

REENTERABLE CODING

USE "GETMAIN" FOR:

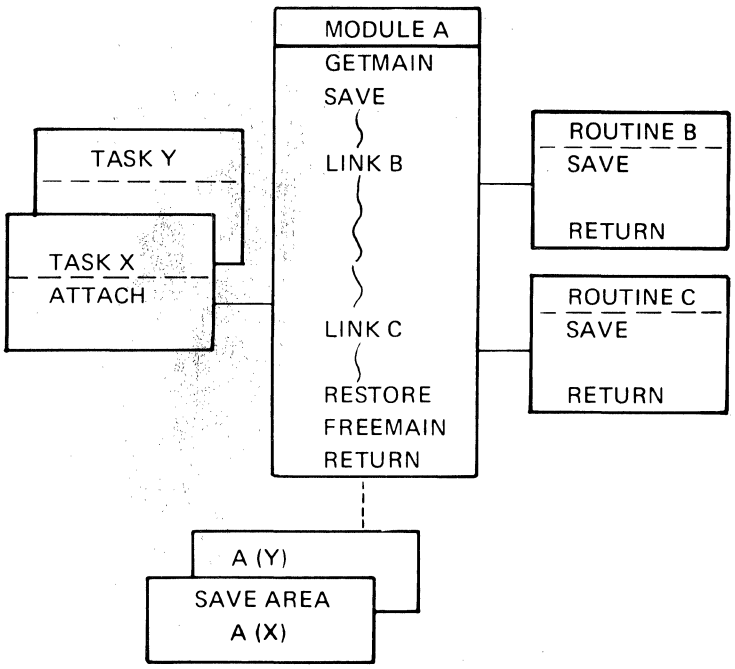
- SAVE AREAS
- WORK AREAS
- SWITCHES
- DCBs
- SYSTEM-MODIFIED MACRO PARAMETER LISTS

CODE MACROS IN:

- REGISTER NOTATION
- LIST AND EXECUTE FORMS

V.1.2

REENTRANT CODE
USE OF SAVE



V.1.3

REENTERABLE SAVE AREA CHAINING

```
PROGNAME  SAVE     (14,12)
          LR       2,15
          USING    PROGNAME,2
          LR       3,1
          GETMAIN  R,LV=72
          ST       13,4(1)
          ST       1,8(13)
          LR       13,1
          . . .
```

V.1.4

GETMAIN MACRO INSTRUCTION

[symbol] GETMAIN EC,LV=number,A=address[,SP=number] [,BNDRY= {DBLWD / PAGE}]

[symbol] GETMAIN EU,LV=number,A=address[,SP=number] [,BNDRY= {DBLWD / PAGE}]

[symbol] GETMAIN VC,LA=address,A=address[,SP=number] [,BNDRY= {DBLWD / PAGE}]

[symbol] GETMAIN VU,LA=address,A=address[,SP=number] [,BNDRY= {DBLWD / PAGE}]

[symbol] GETMAIN RC*,LV=number[,SP=number] [,BNDRY={DBLWD / PAGE}]

[symbol] GETMAIN RU*,LV=number[,SP=number] [,BNDRY={DBLWD / PAGE}]

NOTE: There are several additional forms of the macro. Refer to the reference manual.

*MVS Only

USE OF THE GETMAIN MACRO INSTRUCTION

```
            ...
            GETMAIN     EC,LV=16000,A=ANSWADD     CONDITIONAL REQUEST FOR 16000 BYTES
            LTR         15,15                     TEST RETURN CODE
            BZ          PROCEED1                  IF 16000 BYTES ALLOCATED, PROCEED
            DELETE      EP=REENTMOD               IF NOT, FREE MAIN STORAGE
            GETMAIN     VU,LA=SIZES,A=ANSWADD     ATTEMPT TO GET SMALLER AMOUNT
            L           4,ANSWADD+4               LOAD AND TEST ALLOCATED LENGTH
            CH          4,MIN                     IF 8000 OR MORE, USE PROCEDURE 1
            BNL         PROCEED1                  IF LESS THAN 8000, USE PROCEDURE 2
PROCEED2    ...
PROCEED1    ...
MIN         DC          H'8000'                   MINIMUM SIZE FOR PROCEDURE 1
SIZES       DC          F'4000'                   MINIMUM SIZE TO PROCEED AT ALL
            DC          F'16000'                  SIZE OF AREA FOR MAXIMUM EFFICIENCY
ANSWADD     DC          F'0'                      ADDRESS OF ALLOCATED AREA
            DC          F'0'                      SIZE OF ALLOCATED AREA
```

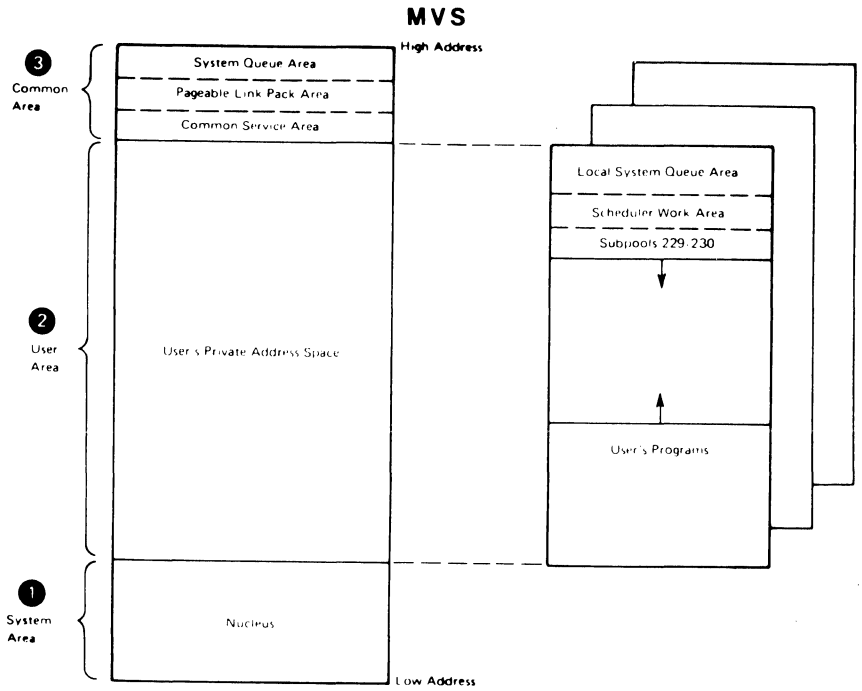
FREEMAIN MACRO INSTRUCTION

```
[symbol]   FREEMAIN   E,LV=number,A=address[,SP=number]
[symbol]   FREEMAIN   L,LA=address,A=address[,SP=number]
[symbol]   FREEMAIN   R,LV=number,A=address[,SP=number]
[symbol]   FREEMAIN   R,LV=number,A=(1) [,SP=number]
[symbol]   FREEMAIN   V,A=address [,SP=number]
```

NOTE: There are several additional forms of the macro. Refer to the reference manual.

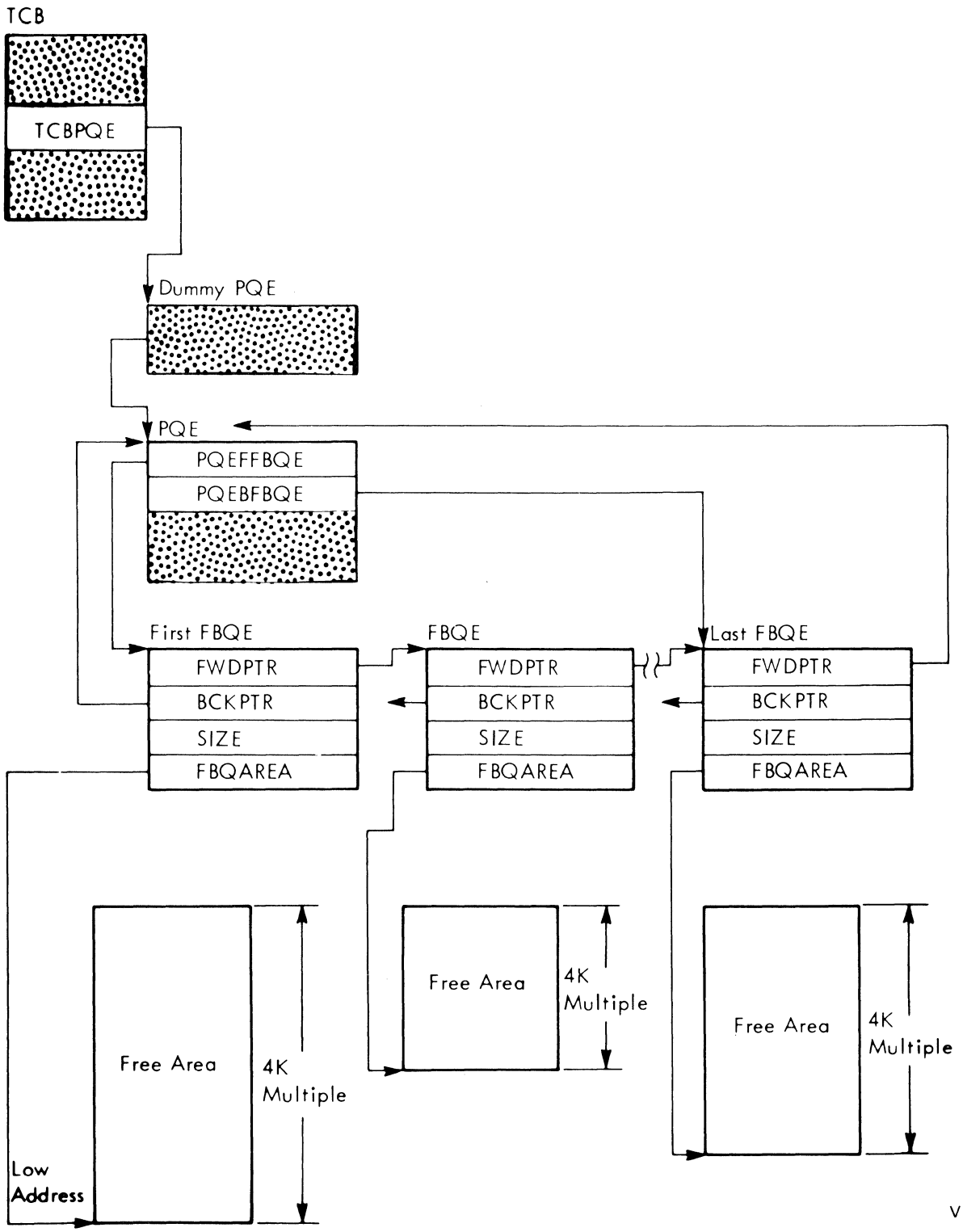
PGRLSE Macro Instruction

```
[symbol]   PGRLSE   LA= { addr1 } ,HA= { addr2 }
                        { (reg1) }     { (reg2) }
```

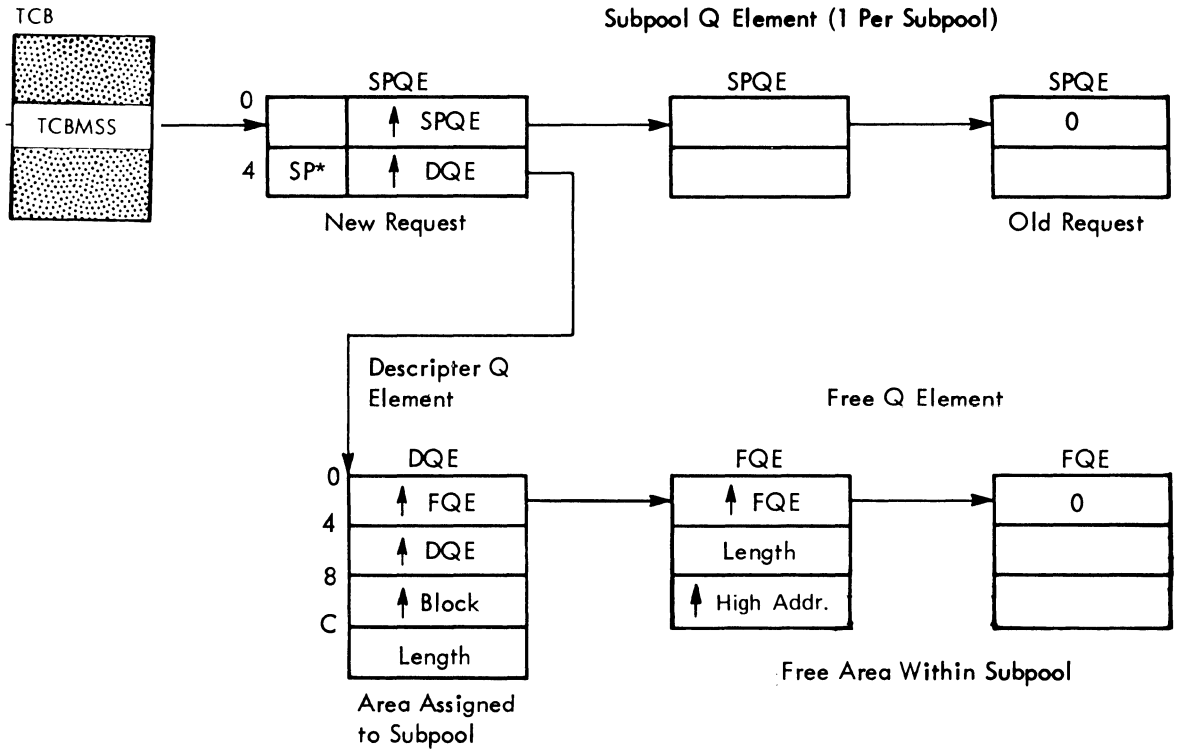


Virtual Storage Layout

VIRTUAL STORAGE CONTROL BLOCKS – VS2

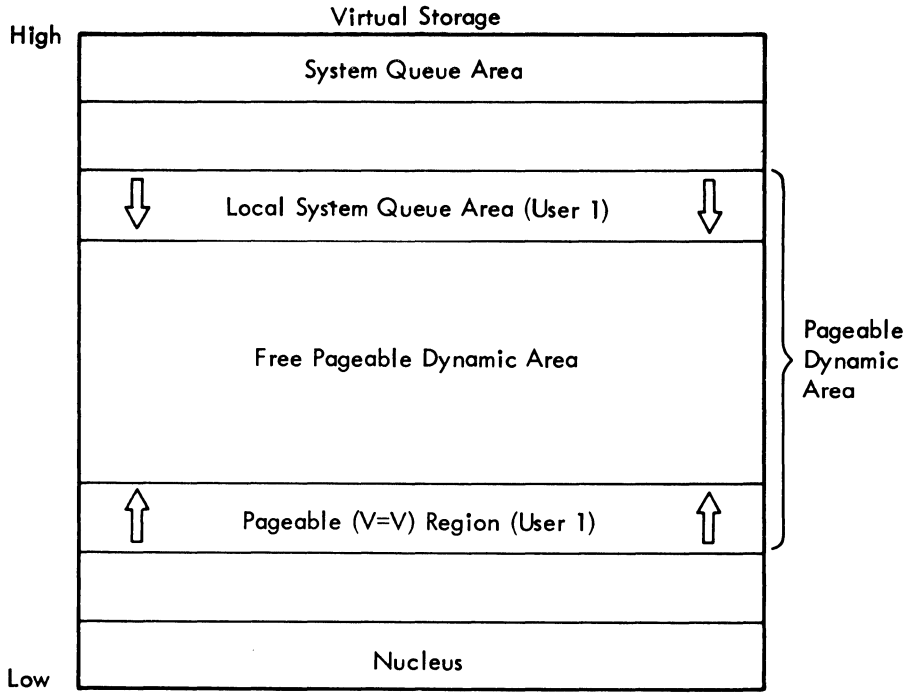


V.2.6



V.2.7

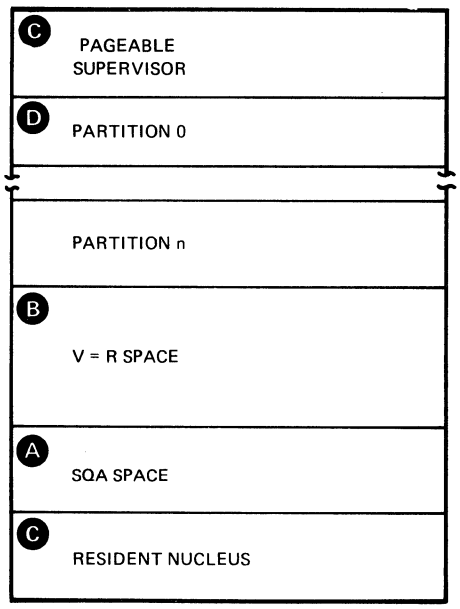
SVS



Note:
⇧⇩ These arrows indicate the direction (high to low or low to high) in which storage addresses are allocated for the respective areas.

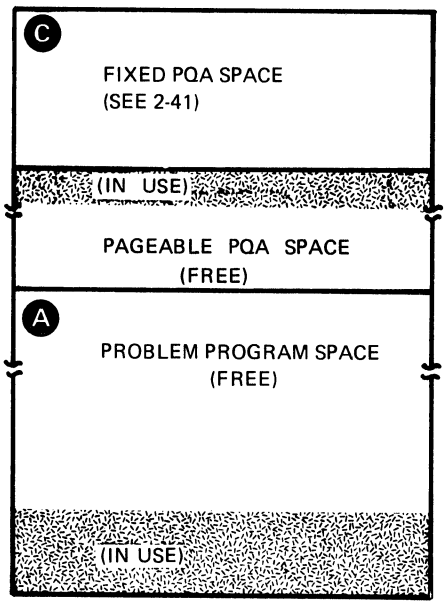
V.2.8

VIRTUAL STORAGE – VS1



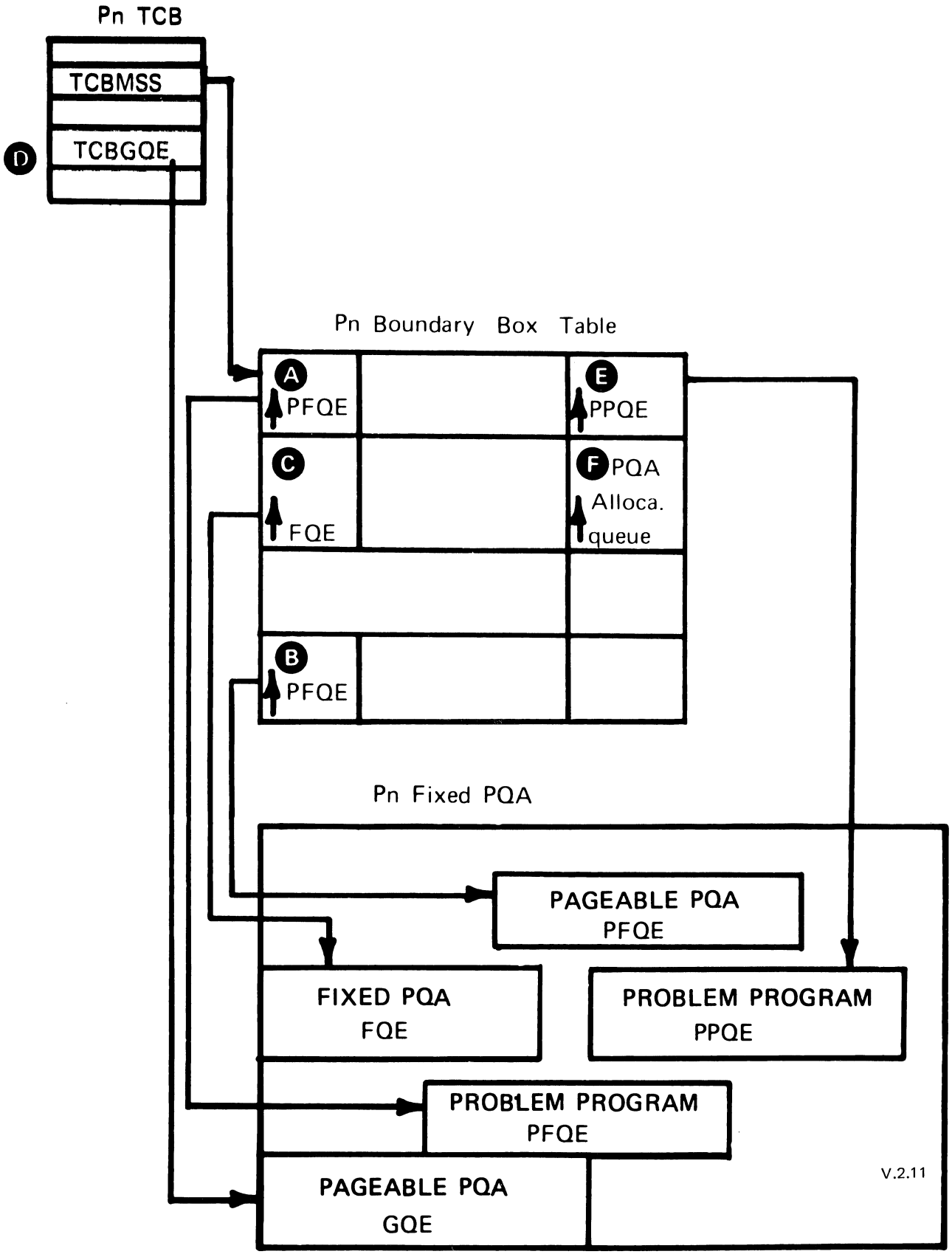
V.2.9

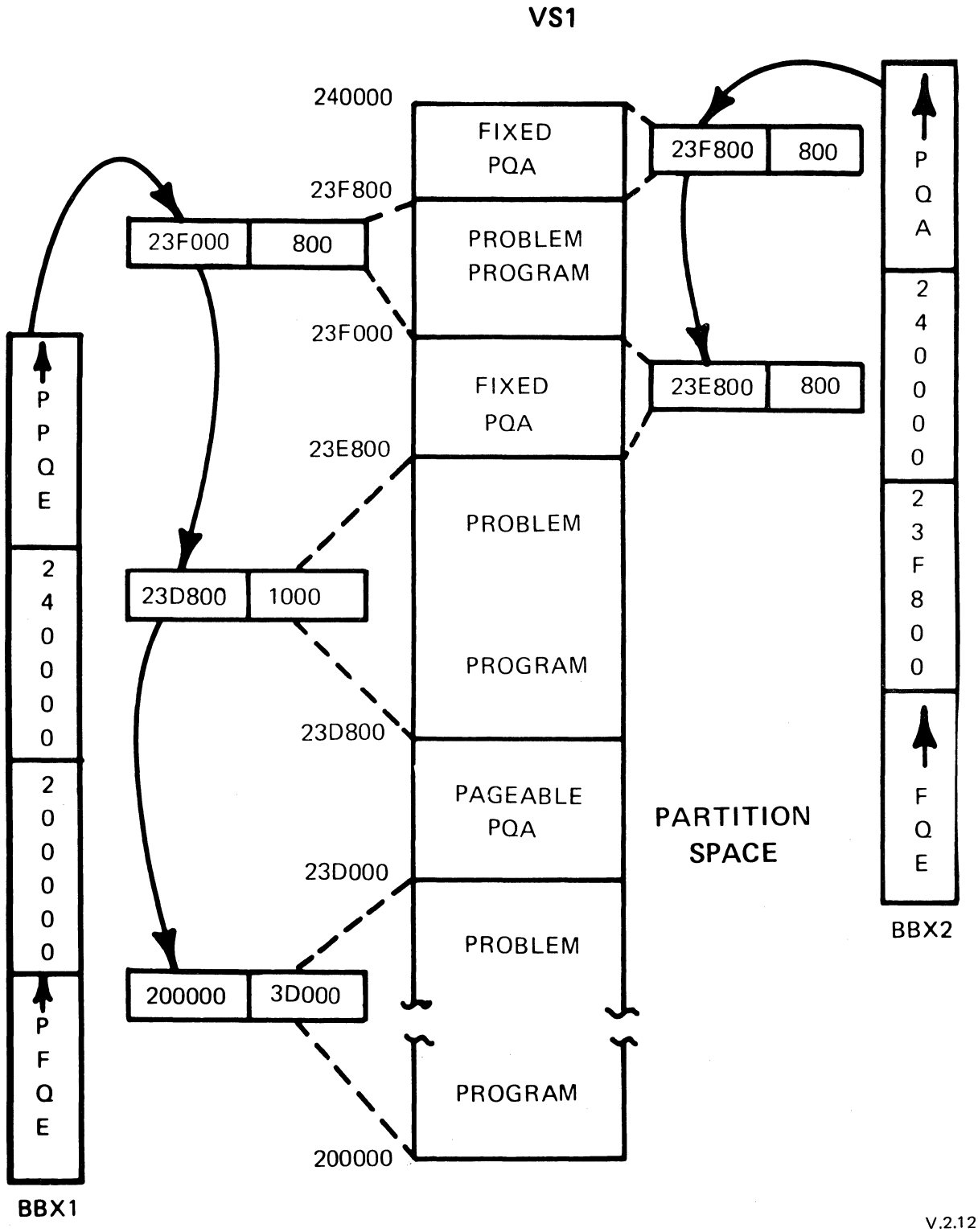
PARTITION n – VS1



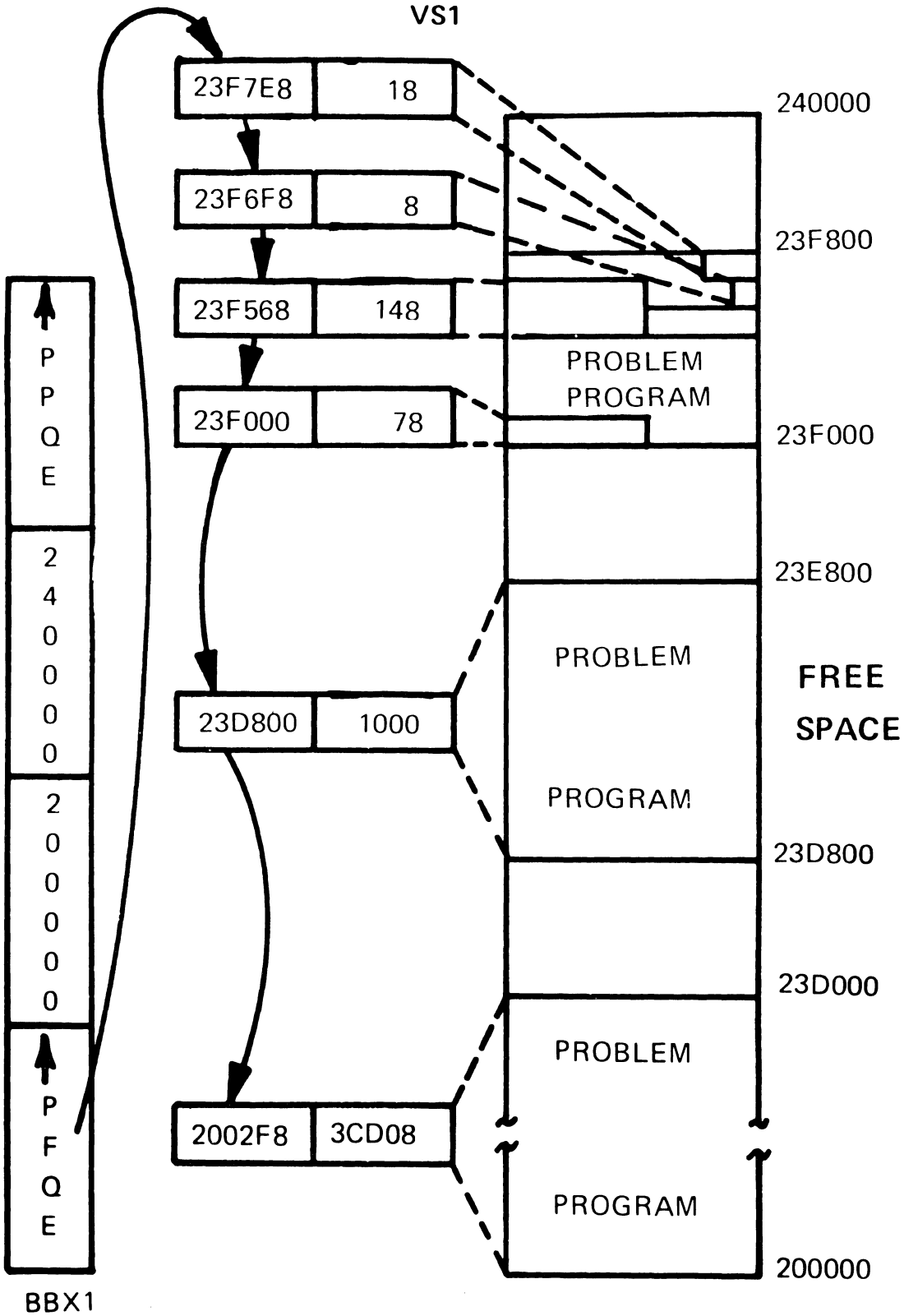
V.2.10

VIRTUAL STORAGE CONTROL BLOCKS – VS1




VS1
240000
23F800
23F000
23E800
23D800
23D000
200000
FIXED PQA
PROBLEM PROGRAM
FIXED PQA
PROBLEM PROGRAM
PAGEABLE PQA
PROBLEM PROGRAM
23F800
800
23E800
800
23F000
800
23D800
1000
200000
3D000
PARTITION SPACE
PQA
240000
23F800
FQE
BBX2
PPQE
240000
200000
PFQE
BBX1

V.2.12

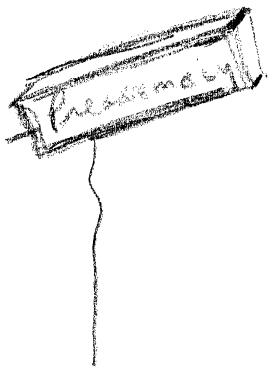

VS1
23F7E8
18
23F6F8
8
23F568
148
23F000
78
23D800
1000
2002F8
3CD08
PPQE
240000
PFQE
BBX1
240000
23F800
PROBLEM
PROGRAM
23F000
23E800
PROBLEM
PROGRAM
FREE
SPACE
23D800
23D000
PROBLEM
PROGRAM
200000

V.2.13

WHY USE MACROS

- SAVE TIME
- LESS PROGRAMMER TRAINING
- EASIER UPDATE
- STANDARDIZATION
- LOCALITY OF REFERENCE

TERMINOLOGY



MACRO DEFINITION - pattern from which statements will be selected

MACRO INSTRUCTION - supplies specific data

MACRO GENERATION - process of tailoring macro for the specific data supplied

PRE-ASSEMBLY - time when assembler processes macros

MACRO DEFINITION

HEADER MACRO
PROTOTYPE & OPERANDS
BODY OF MACRO MODEL STMTS
TRAILER MEND

V.3.3

MACRO and MEND STATEMENTS

name (Label)	operation	operand
must be blank	MACRO	not required (Not listed)
blank or sequence symbol (Internal Symbol)	MEND (Exit)	not required (Not listed)

V.3.4

Prototype Model for Macro Inst.

& NAME	OPCODE	&OP1,&OP2, . . . optional	COMMENTS
& NAME	GET	&DCB,&WORK	- direct substitution

Consider scope of symbol — local or global.

V.3.5

VARIABLE SYMBOLS

&AXXXXXX

&FLD1
&MULTPLR
&DIVDND

SYMBOLIC PARAMETERS - variable symbols declared in the prototype statement of a macro definition

V.3.6

KEYWORD PROTOTYPE

Only in operand fields

```
&NAM     MAC2     &OPER1=,&OPER2=2,&OPER3=ABC

INSTRUCTIONS

XYZ      MAC2          OPER1=FLDA

                USES   &OPER1=FLDA
                       &OPER2=2
                       &OPER3=ABC

ABC      MAC2          OPER2=4,OPER3=RST

                USES   &OPER1=null character string
                       &OPER2=4
                       &OPER3=RST
```

V.3.7

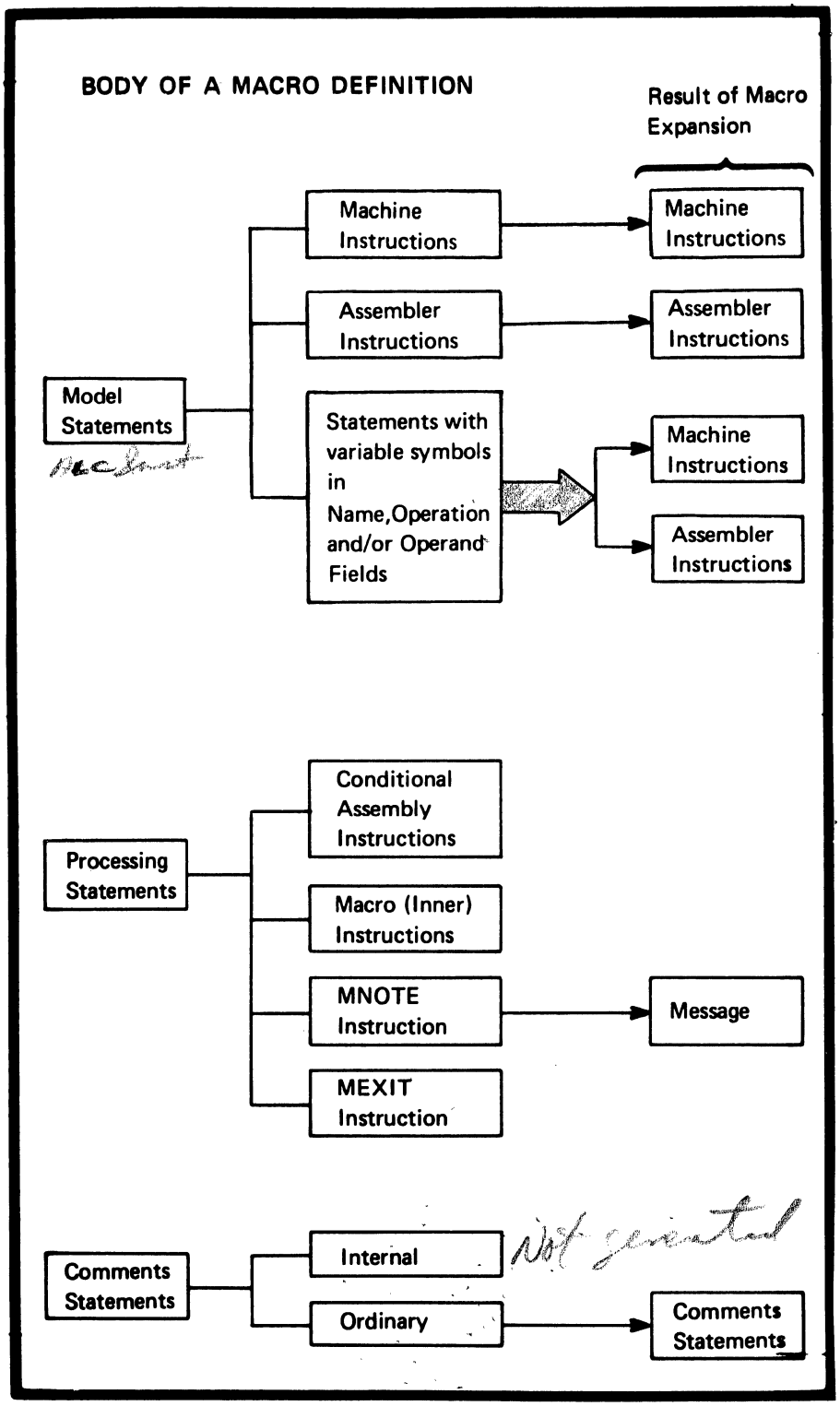
POSITIONAL and KEYWORD OPERANDS

default value

```
&NAME    MCRO     &OP1,&OP2=7,&OP3,&OP4=

same     MCRO     FLDA,OP2=6,FLDC,OP4=R
         MCRO     FLDA,FLDC,OP4=R,OP2=6
         MCRO     OP2=6,OP4=R,FLDA,FLDC
         MCRO     OP4=R,FLDA,OP2=6,FLDC

                  &OP1=FLDA; &OP2=6; &OP3=FLDC; &OP4=R

not same MCRO     FLDC,OP2=6,FLDA,OP4=R

                  &OP1=FLDC; &OP2=6; &OP3=FLDA; &OP4=R
```

Positional first, keywords second

V.3.8


BODY OF A MACRO DEFINITION
Result of Macro Expansion
Model Statements
Machine Instructions
Assembler Instructions
Statements with variable symbols in Name,Operation and/or Operand Fields
Machine Instructions
Assembler Instructions
Machine Instructions
Assembler Instructions
Processing Statements
Conditional Assembly Instructions
Macro (Inner) Instructions
MNOTE Instruction
MEXIT Instruction
Message
Comments Statements
Internal
Ordinary
Comments Statements

MODEL STATEMENTS

```
         MACRO
         MAC2    &FLDA, &FLDB, &FLDC, &FLDD
         LA      3,4(3)          machine instruction
         LA      5, &FLDC        machine instruction
         MP      &FLDA,&FLDB     with variable symbols
&FLDC    DC      C'&FLDD'        assembler instruction
         MEND
```

```
         MAC2    FLD1,FLD2,FLD3,XYZ
+        LA      3,4(3)
+        LA      5,FLD3
+        MP      FLD1,FLD2
+FLD3    DC      C'XYZ'
```

V.3.10

INSTRUCTION GENERATION

at assembly time

MODEL STMTS of MACRO DEFN	GENERATED INSTRUCTIONS
can contain	must contain
ORDINARY SYMBOLS	VALID OP CODES
VARIABLE SYMBOLS	OPERANDS
Combination of ORDINARY and VARIABLE SYMBOLS	LABELS

V.3.11

MNOTE

MAX severity Error code.

Forms

1) MNOTE n,'error message'

$0 \leqslant n \leqslant 255$

Contribute to assembler return code

2) MNOTE ,'error - default severity code'

n=1

3) MNOTE *,'comment - not an error'

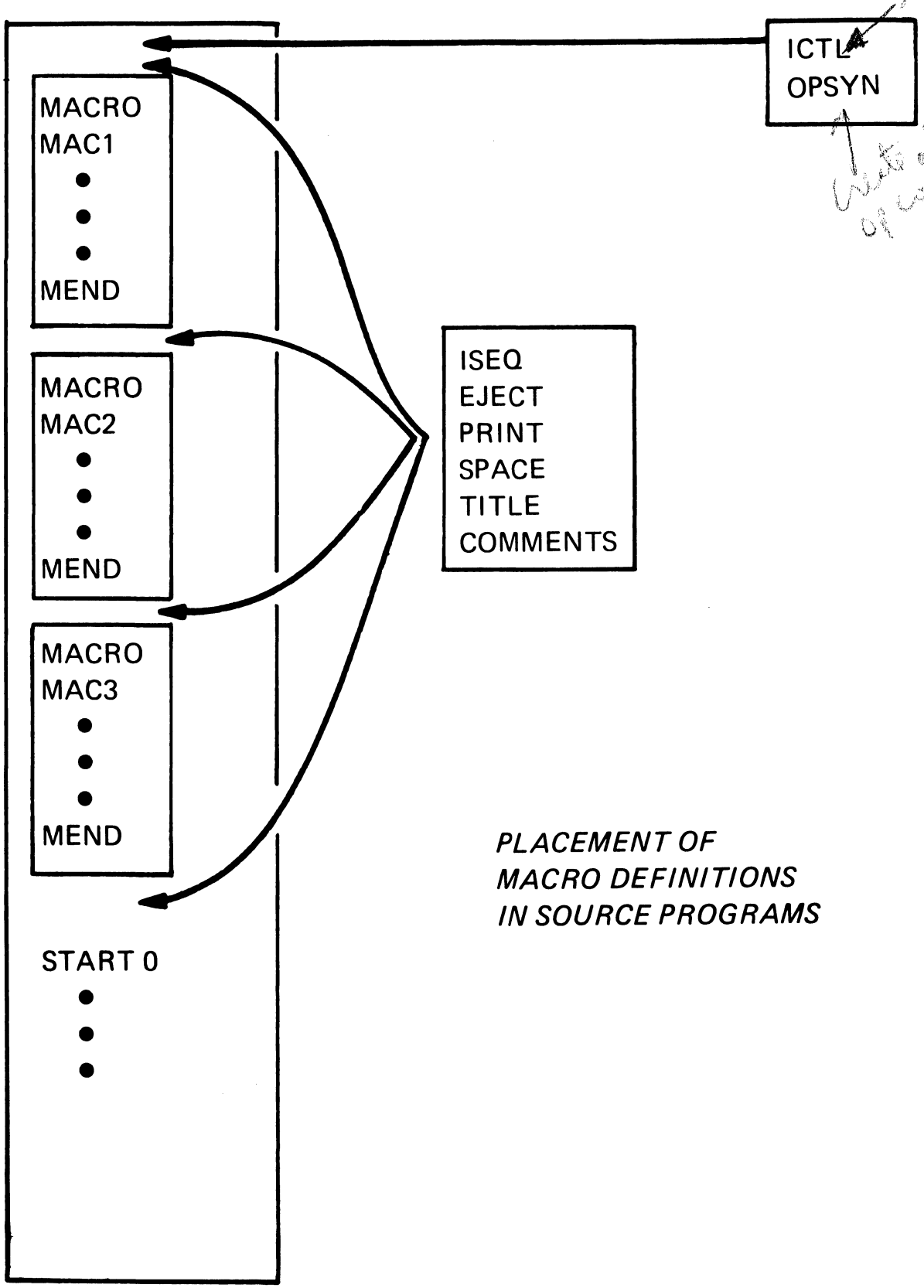
4) MNOTE 'comment only'

MNOTE EXAMPLES

```
MNOTE     3, 'INCORRECT LENGTH'

MNOTE     3, 'INCORRECT LENGTH FOR &FLDA'
                                  will be replaced by
                                  corresponding field in
                                  macro instruction

MNOTE     ,'GENERATION WAS TERMINATED'
            severity code of 1
            MGXIT

MNOTE     'RESULT WILL BE PLACED IN &RESULT4'
            AGO     .SEQ

.SEQ      MLWD
```


COMMENTS

```
.* THIS COMMENT WILL NOT APPEAR ON SOURCE LISTING

* THIS COMMENT WILL PRINT

* &FLDA WILL NOT BE REPLACED BY CORRESPONDING OPERAND
```



PLACEMENT OF MACRO DEFINITIONS IN SOURCE PROGRAMS

ADD MACRO TO SYS1.MACLIB

```
//ADDMAC    JOB
//STEP1     EXEC    PGM=IEBUPDTE,PARM=MOD
//SYSUT1    DD      DSN=SYS1.MACLIB,DISP=OLD
//SYSUT2    DD      DSN=SYS1.MACLIB,DISP=OLD
//SYSPRINT  DD      SYSOUT=A
//SYSIN     DD      DATA
./          ADD     LIST=ALL,NAME=NEWMAC
            MACRO
            NEWMAC      &OP1, &OP2
            .
            .
            .
            MEND
/           ENDUP
/*
```

V.3.15

POSITIONAL MACRO -- DIRECT SUBSTITUTION

```
         MACRO
&NAM     ADD1    &F1, &F2, &R, &P1, &P2
&NAM     PACK    &P1, &F1
         PACK    &P2, &F2                macro
         AP      &P1, &P2              definition
         UNPK    &R, &P1
         MEND

 ADD1    ADD1    A,B,C,D,E        macro instruction
+ADD1    PACK    D,A
+        PACK    E,B                generated
+        AP      D,E                code
+        UNPK    C,D

 A       DC      C'123'
 B       DC      C'456'
 C       DS      CL3
 D       DS      CL2
```

V.3.16

KEYWORD MACRO -- DIRECT SUBSTITUTION

```
          MACRO
&NAME     MULT1     &FLD=, &NUM=2, &RESULT=
&NAME     PACK      &RESULT, &FLD
          MP        &RESULT,=P'&NUM'
          MNOTE     '&FLD MULTIPLIED BY &NUM'
.* &FLD WAS MULTIPLIED BY &NUM
.* 07/02/73
        MEND
        START     ϕ
          .
          .
          .
  TWO   MULT1     FLD=A,RESULT=B
+ TWO   PACK      B,A
+       MP        B,=P'2'
+ A MULTIPLIED BY 2
+* &FLD WAS MULTIPLIED BY &NUM
          .
          .
          .

 A      DC        C'123'
 B      DS        PL3
          .
          .
          .
```

ATTRIBUTES

- TYPE
- LENGTH
- SCALING
- INTEGER
- COUNT
- NUMBER

V.3.18

TYPE ATTRIBUTE

```
FLDA     DC      C'1234'
FLDB     DC      F'14'
FLDC     DS      H
FLDD     DC      A(FLDA)
CCW1     CCW     X'05',*,X'00',4
MAC1     ABEND     001
```

TYPE OF

```
FLDA     EQU     C
FLDB     EQU     F
FLDC     EQU     H
FLDD     EQU     A
CCW1     EQU     W
MAC1     EQU     M
```

V.3.19

USE OF TYPE ATTRIBUTE

conceptual

DATA TYPE
IF T'&FLDA IS NOT 'P'
- WRITE ERROR MESSAGE AND
TERMINATE GENERATION

OR - BRANCH TO CODE TO PACK DATA

MISSING OPERAND
IF T'&OP2 IS 'O'
- WRITE COMMENT AND USE DEFAULT

OR - WRITE ERROR MESSAGE AND
TERMINATE GENERATION

'O' with
N
V

VALUE SUBSTITUTION
IF T'FLD IS 'H'
- MODIFY CODE TO REFLECT
HALFWORD INSTRUCTIONS

LENGTH ATTRIBUTE

FLDA	DC	C'1234'
FLDB	DS	H
FLDC	DC	FL2'14'
MAC	ABEND	002
INST	LR	3,4

LENGTH OF

FLDA	EQ	4
FLDB	EQ	2
FLDC	EQ	2
MAC	EQ	0
INST	EQ	2

USE OF LENGTH ATTRIBUTE

VALID LENGTH

```
IF L'&MULT    > 8
     - WRITE ERROR MESSAGE AND
          TERMINATE GENERATION
```

DETERMINATION OF FIELD SIZE

```
ADD L'&MULT and L'&MCAND TO
   DETERMINE SIZE REQUIRED FOR
   PRODUCT
```

SUBSTITUTION INTO LENGTH FIELD

```
FLDA     DS     CL(L'DATA)
MVC        FLDA(L'&FLDB),&FLDB
```

V.3.22

COUNT ATTRIBUTE

```
MACRO1     &OP1,&OP2,&OP3,&OP4,&OP5, &OP6


MACRO1     OPERAND1,OP2,,2,(14), (1,2,3,4,5)
                                 (6,2,3,,5)

K'&OP1     EQ     8
K'&OP2     EQ     3
K'&OP3     EQ     0
K'&OP4     EQ     1
K'&OP5     EQ     4
N'&OP6     EQ     5
```

V.3.23

ATTRIBUTES

	PURPOSE	MAIN USES
TYPE *T'*	IDENTIFIES THE TYPE OF DATA REPRESENTED	- TESTS FOR DATA TYPE - VALUE SUBSTITUTION - CHECK FOR MISSING OPERAND
LENGTH *L'*	NUMBER OF BYTES THAT DATA OCCUPIES	- SUBSTITUTION INTO LENGTH FIELDS - COMPUTATION OF STORAGE REQUIREMENTS
SCALING *S'*	REFERS TO DECIMAL PT. POSITION	- TESTING AND REGULATING DECIMAL PT. POSITION - SUBSTITUTION INTO A SCALE MODIFIER
INTEGER *I'*	A FUNCTION OF LENGTH AND SCALING ATTRIBUTES	- TO KEEP TRACK OF SIGNIFICANT DIGITS
COUNT *K'*	NUMBER OF CHARACTERS REQUIRED TO REPRESENT DATA	- SCANNING CHARACTER STRINGS - INDEXES FOR SUBSTRINGS
NUMBER *N'*	NUMBER OF SUBLIST ENTRIES IN A MACRO INSTRUCTION OPERAND	- FOR SCANNING SUBLISTS

```
                MACRO
                  .
                  .
                  .
                AGO          .BRADDR
.BACK           ANOP
                  .
                  .
                  .
                MEXIT
.BRADDR         ANOP
                  .
                  .
                  .
                  .
                  .
                AGO          .BACK

                MEND
```

UNCONDITIONAL BRANCHING

AIF - Conditional Branch

```
AIF     (T'&FLDA NE    'P') . PACK
                test to see if code is to
                be generated

AIF     (&COUNTER EQ 10).ENDLOOP
                loop control

AIF     (T'&FLD EQ 'O').ERROR1
                test for error condition

AIF     ('&FLDX' EQ 'NE').NOTEQ
                test for actual value
```

EDITING CAPABILITIES

```
         MACRO
&NAM     ADD1     &F1,&F2,&R,&P1,&P2
         AIF      (T'&F1 EQ 'O').MISSOP
         AIF      (T'&F2 EQ 'O').MISSOP
         AIF      (T'&R EQ 'O').MISSOP
         AIF      (T'&P1 EQ 'O').MISSOP
         AIF      (T'&P2 NE 'O').CHKF1
.MISSOP  MNOTE    4,'MISSING OPERAND'
         MEXIT
.CHKF1   AIF      (T'&F1 EQ 'Z').CHKF2
         MNOTE       4, '&F1 NOT ZONED'
         MEXIT
.CHKF2   AIF      (T'&F2 EQ 'Z').GENERAT
         MNOTE       4,'&F2 NOT ZONED'
         MEXIT
.GENERAT    ANOP
&NAM        PACK     &P1,&F1
            PACK     &P2,&F2
            AP       &P1,&P2
            UNPK     &R,&P1
            MEND
```

USES OF SET SYMBOLS

```
        MACRO
          .
          .
          .
.LOOP

        Add 1 to a counter
          .
          .
        AIF   (&COUNT LE 10).LOOP
          .
          .
          .

        Set switch to 1 if &OP1 missing
        MNOTE ,'OP1 MISSING'
          .
          .
        Set switch to 1 if &OP2 missing
        MNOTE ,' OP2 MISSING'

        AIF   (&SWITCH EQ 0).CONT
        MEXIT
.CONT
```

DECLARATION OF SET SYMBOLS

```
LCLA    &A1,&A2,&A3,&A4
LCLB    &SW1,&SW2
LCLC    &CHAR1,&CHAR2
```

0	0	0	0
&A1	&A2	&A3	&A4

0	0
&SW1	&SW2

null	null
&CHAR1	&CHAR2

V.3.28

Examples of Arithmetic Expressions

```
+ — — & A
& A + — & B
& A + * & B                INVALID
* & A                      INVALID
& A & B                    INVALID
2(10 + & C)                INVALID
& A + & B * & C * (10 — & D)
& A     +     & B          INVALID
& A + B '101' * 2 — (& C/2)
```

V.3.30

SETA STATEMENTS

declare set symbol &A1(20)

```
      LCLA     &A1, &A2, &A3, &A4
```

arith

```
&A1   SETA     10
&A2   SETA     &A1
&A3   SETA     K'&FLDA
&A4   SETA     2*K'&FLDA
&A3   SETA     &A3–1
```

how-1

LOOP CONTROL

```
            MACRO
            FULLDC   &NUMBER
            LCLA     &COUNTER
.LOOP       ANOP
&COUNTER    SETA     &COUNTER+1
            AIF      (&COUNTER GT &NUMBER).END
            DC       F'0'
            AGO      .LOOP
.END        MEND
```

generate a DC

SETC FORMATS

```
        LCLC    &C1,&C2,&C3,&C4,&C5
&C1     SETC    T'&FIELD          type attribute
&C2     SETC    'ABC'             character expression
&C3     SETC    'ABCDE'(1,3)      substring
&C4     SETC    'ABC'.'DEF'       concatenation
                  or
                'ABC'.'ABCDEF'(4,3)
```


SETC

	VALUE OF &A1	VALUE OF &C1
&C1 SETC '&A1' where &A1 is a SETA symbol	−1	1
	+1	1
	0	0
	002	2
&C1 SETC '−1'		−1
&C1 SETC '002'		002
&C1 SETC '&A1+4'	−25	25+4
&C1 SETC '&&A1'		&A1

SET B

	LCLB	&B1, &B2, &B3	
&B1	SET B	0	binary value
&B2	SET B	(1)	binary value in parenthesis
&B3	SET B	(T'&FLD EQ 'F')	logical expression

LOGICAL EXPRESSION

can be

arithmetic exp { EQ, NE, GT, LT, GE, LE } arithmetic exp

or

character comparand { EQ, NE, GT, LT, GE, LE } character comparand

or

SET B variable symbol

or

0

or

1

or a combination using logical operators – AND, OR, NOT

(&A GT 100 OR '&C' EQ 'R')	FALSE
(T'&OP1 EQ 'F' AND T'&OP2 EQ 'F')	TRUE
(&COUNT LT 10 OR NOT (T'&OP1 EQ T'&OP2))	TRUE

where

value of	is
&A	50
&C	S
T'&OP1	F
T'&OP2	F
&COUNT	7

LOGICAL EXPRESSIONS


```
                MACRO          BINARY SWITCH

&NAME           ADD 5          &OP1, &OP2, &OP3, &OP4, &OP5
                LCLB           &SWITCH
                AIF            (T'&OP1 NE 'O').CHK2
                MNOTE          ,'OPERAND 1 MISSING'
&SWITCH         SETB           1
.CHK2           AIF            (T'&OP2 NE 'O').CHK3
                MNOTE          ,'OPERAND 2 MISSING'
&SWITCH         SETB           1
.CHK3           AIF            (T'&OP3 NE 'O').CHK4
                MNOTE          ,'OPERAND 3 MISSING'
&SWITCH         SETB           1
.CHK4           AIF            (T'&OP4 NE 'O').CHK5
                MNOTE          ,'OPERAND 4 MISSING'
&SWITCH         SETB           1
.CHK5           AIF            (T'&OP5 NE 'O'). OK
                MNOTE          ,'OPERAND 5 MISSING'
&SWITCH         SETB           1
.OK             AIF            (&SWITCH NE 1).GEN
                MEXIT
.GEN            ANOP
&NAME           PACK           &OP4, &OP1
                PACK           &OP5, &OP2
                AP             &OP4, &OP5
                UNPK           &OP3, &OP4
                MEND
```

GLOBAL SYMBOLS BETWEEN MACRO AND OUTSIDE

```
            MACRO
&NAME       LOADA       &WD
            GBLA        &A
&A          SETA        4
&NAME       L           &A,&WD
&A          SETA        &A+1
            MEND

            GBLA        &A
ONE         CSECT
FIRST       LOADA       NUM
            L           &A,NEW
            .
            .
            END
```

Generated Code

```
FIRST       LOADA       NUM
+FIRST      L           4,NUM
            L           5,NEW
            .
            .
```

V.3.39

GLOBAL SYMBOLS BETWEEN MACROS

```
            MACRO
&NAME       LOADA       &WD
            GBLA        &A
            AIF         (&A GE 4).NEW
&A          SETA        4
.NEW        ANOP
&NAME       L           &A,&WD
&A          SETA        &A+1

            MEND

            MACRO

            LOADB       &WD
            GBLA        &A
            AIF         (&A GE 4).NEXT
&A          SETA        4
.NEXT       ANOP
            L           &A,&WD
&A          SETA        &A+1
            MEND

FIRST       LOADA       BEGIN
+FIRST      L           4,BEGIN
            LOADB       NEXT
+           L           5,NEXT
            LOADA       FULL
+           L           6,FULL
            LOADB       TOTAL
+           L           7,TOTAL
            END         FIRST
```

V.3.40

MACRO

```
        MOVE    &TO, &FROM
        MVC     &TO(1), &FROM(1)
        MVC     &TO(2), &FROM(2)
        MVC     &TO(3), &FROM(3)
        MVC     &TO(4), &FROM(4)
        MVC     &TO(5), &FROM(5)
        MEND
```

SUBLISTS and SUBSCRIPTS

```
        MOVE    (A1,B1,C1,D1,E1), (A2,B2,C2,D2,E2)
+       MVC     A1,A2
+       MVC     B1,B2
+       MVC     C1,C2
+       MVC     D1,D2
+       MVC     E1,E2
```


ADD A VARIABLE NUMBER OF FIELDS

```
        MACRO
        ADD     &REG, &FLDS
        LCLA    &COUNTER
&COUNTER SETA   1
.LOOP   ANOP
        AIF     (T'&FLDS(&COUNTER) NE 'F').ERR
        A       &REG, &FLDS(&COUNTER)
&COUNTER SETA   &COUNTER + 1
        AIF     (&COUNTER LE N'&FLDS).LOOP
        AGO     .END
.ERR    MNOTE   4,'FIELDS MUST BE FULLWORD'
.END    MEND
```

SUBSCRIPTED SET SYMBOLS

```
LCLA   &A(5)
LCLB   &B(8)
GBLC   &C(4)
```

0	0	0	0	0

0	0	0	0	0	0	0	0

null	null	null	null

```
&A(3)   SETA   34
&B(1)   SETB   1
&C(5)   SETC   'ABC'     INVALID
```

0	0	34	0	0

1	0	0	0	0	0	0	0

null	null	null	null

V.3.43

Concatenation

		RESULT
&A &B	&A has value FLD &B has value B	FLDB
&A.X		FLDX
SAVE &A		SAVE FLD
&A.(2)		FLD(2)
P'&I..&F'	&I has value 9 &F has value 5	P'9.5'
P'&I.&F'		P'95'
P'&I&F'		P'95'

V.3.44

MACRO

```
&NAME     COMP     &A,&B,&BR,&REG
          LCLC     &TY
          AIF      (T'&A EQ T'&B).TYPEQ
          MNOTE    4,'OPERANDS INVALID TYPE'
          MEXIT
.ERR      MNOTE    4,'MUST BE F OR H'
          MEXIT
.TYPEQ    AIF      (T'&A EQ 'F').FULL
          AIF      (T'&A NE 'H').ERR
 &TY      SET C    'H'
.FULL     ANOP
&NAME     L&TY     &REG,&A
          C&TY     &REG, &B
          BC       8, &BR
          MEND
```

CONCATENATION EXAMPLE

V.3.45

DATE and TIME

```
MACRO
DATETIME
MNOTE     *,'DATE     &SYSDATE - - - TIME &SYSTIME'
MEND
START     0
DATETIME
+ DATE    07/12/73 - - - TIME 11.20
```

V.3.46

&SYSPARM

```
//STEP  EXEC  ASMFC,PARM=SYSPARM(TEST)

        MACRO
        MAC1
        AIF     (&SYSPARM NE 'TEST').SKIP
        MNOTE   * ,'TEST    &SYSDATE'
.SKIP   ANOP
        MEND
```

&SYSECT

```
        MACRO
        MAC2
          ~
        DC          A(&SYSECT)
          ~
        MEND

BEGIN START         0
          ~
        MAC2
          ~
+       DC          A(BEGIN)
          ~
```

USE OF &SYSECT

```
                    MACRO

                    MACI
                     ~
TWO                 CSECT  (Macro needs to generate its own CSECT
                     ~      for some reason)
&SYSECT             CSECT  (Before macro-generation is done, it is
                     ~      necessary to "restore" user's CSECT, so
                    MEND    code AFTER the macro call will be in
                            same CSECT as before)
```

```
PROG1               START 0

                    MACI
+                    ~
+
+TWO                CSECT
+                    ~
+
+PROG1               CSECT
                     ~
```


&SYSLIST

only to positional parms.

NAME	MAC6	FLD1,FLD2,,(A,B,C),FLD5	macro instruction

&SYSLIST(2)	FLD2
&SYSLIST(3)	null
&SYSLIST(4,1)	A
&SYSLIST(6)	null
&SYSLIST(0)	NAME
N' &SYSLIST	5
N' &SYSLIST(4)	3

&SYSNDX

```
                 MACRO
                 MAC 8      &P1, &P2
                   .
                   .
&P1 &SYSNDX      DC         F'&P2'
AREA&SYSNDX      DS         F
                   .
                   .
                   .
                 MEND

                 START      0
                                          &SYSNDX
                                            is
                                           0000
                 MAC8       TWO,2
                   .
                   .
 + TWO0001       DC         F'2'
 + AREA0001      DS         F
                   .
                   .

                 MAC8       TWO,20

 + TWO0002       DC         F'20'
 + AREA0002      DS         F
```

V.3.50

NESTED MACROS

```
        MACRO
        OUTER   &P1, &KEY1=,&P2
        LCLC    &C
&C      SETC    'ABC'
          .
          .
          .
        INNER   &P1,&KEY1,&C
          .
          .
          .
        MEND
        MACRO
        INNER   &A, &B, &C
        L       3, &A
          .
          .
&B      DC      C' &C'
          .
          .
        MEND
        START   0
        OUTER   FLDA, KEY1=FLDB, FLDC
          .
          .
+       L       3, FLDA
          .
          .
          .
+FLDB   DC      C'ABC'
          .
          .
```

PROGRAM STRUCTURE

SIMPLE

- SINGLE LOAD MODULE
- LOADED INTO CORE AS AN ENTITY

PLANNED OVERLAY

- SINGLE LOAD MODULE
- SEGMENTS OTHER THAN ROOT SEGMENT BROUGHT INTO CORE WHEN NEEDED
- ALL SEGMENTS WERE PROCESSED BY THE LINKAGE EDITOR AT THE SAME TIME

DYNAMIC SERIAL

- MULTIPLE LOAD MODULES
- EACH LOAD MODULE WAS PROCESSED BY THE LINKAGE EDITOR SEPARATELY
- COMMUNICATE BY PASSING PARAMETERS
- SUBPROGRAMS LOADED WHEN NEEDED INTO DYNAMIC LOCATIONS

DYNAMIC PARALLEL

- PARALLEL EXECUTION OF SUBPROGRAMS

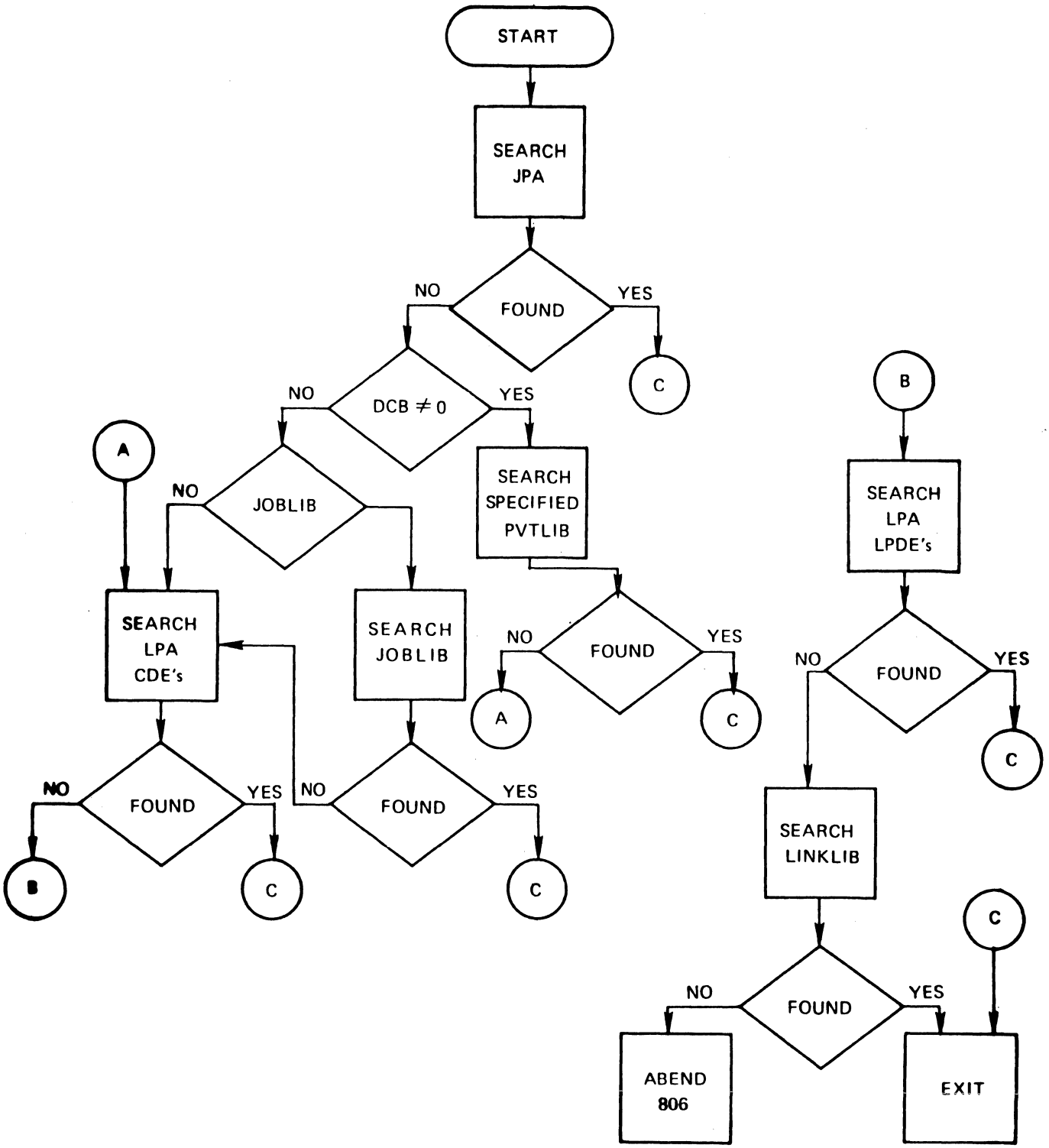
V.4.1

PROGRAM DESIGN FACTORS

QUESTION	VS ANSWER
REPETITIVE ROUTINES?	PROGRAM ONCE AS SUBROUTINE. WILL BE HANDLED AUTOMATICALLY BY SYSTEM.
TOO LITTLE MAIN STORAGE? TOO LARGE A PROGRAM?	OVERLAY OR DYNAMIC STRUCTURE.
MULTIPLE USAGE? (SERIAL OR CONCURRENT)	REUSABLE AND REENTERABLE MODULES.
NOT MUCH PROGRAMMING TIME?	SEGMENTIZE PROGRAM. USE SEVERAL PROGRAMMERS. REASSEMBLE PROGRAM AT USER'S CHOICE OF SOURCE, LINK EDIT, OR EXECUTE TIME.
MORE THAN ONE LANGUAGE NEEDED?	LINKAGE EDITOR.

V.4.2

MODULE SEARCH LOGIC



LOAD and DELETE MACRO INSTRUCTIONS

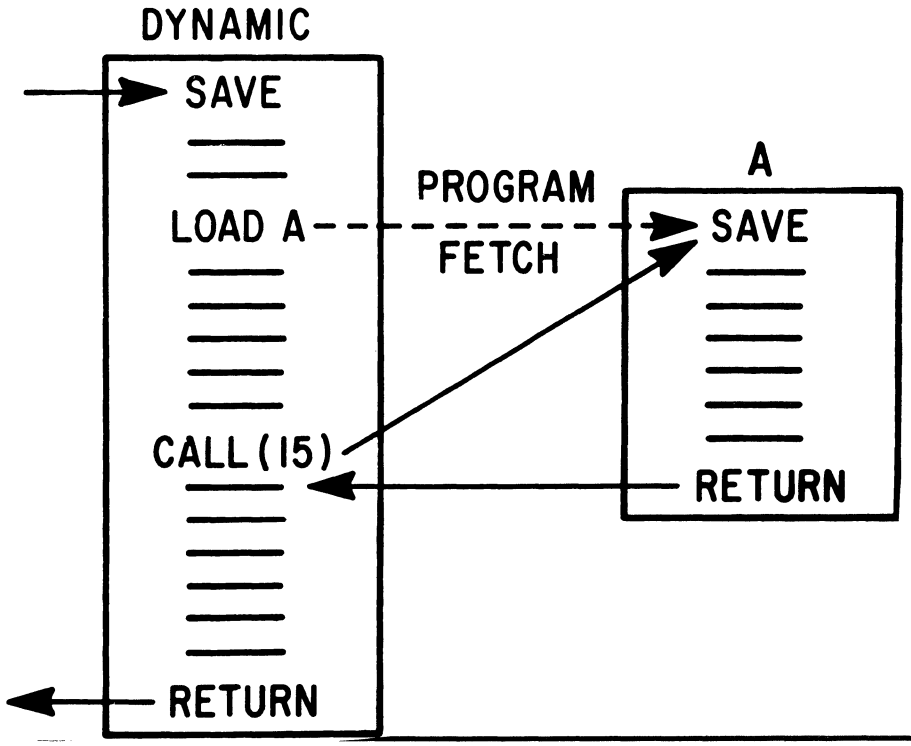
[symbol] LOAD EP=symbol[,DCB=dcb address]

[symbol] DELETE EP=symbol

NOTE: There are several additional forms of the macros. Refer to the reference manual.

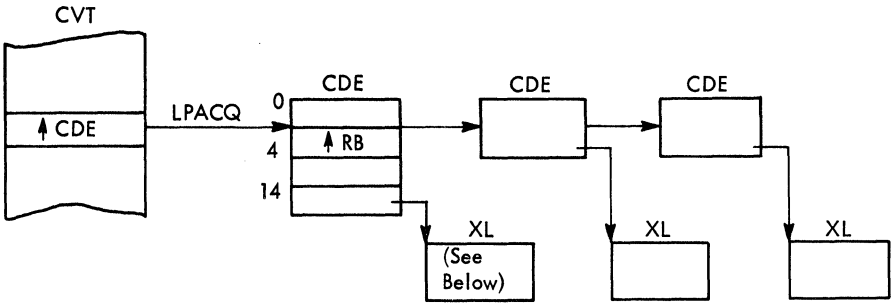
V.4.4

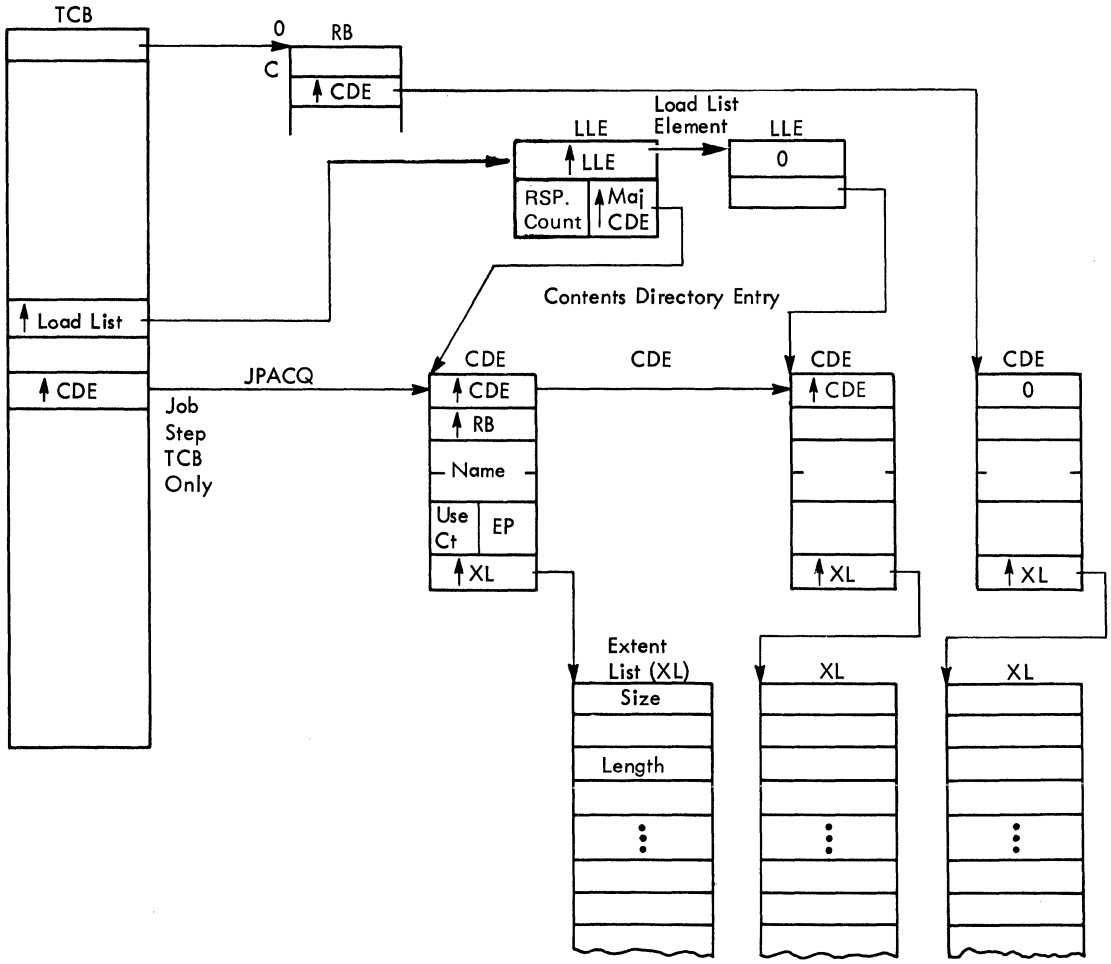
THE LOAD OPERATION



V.4.5

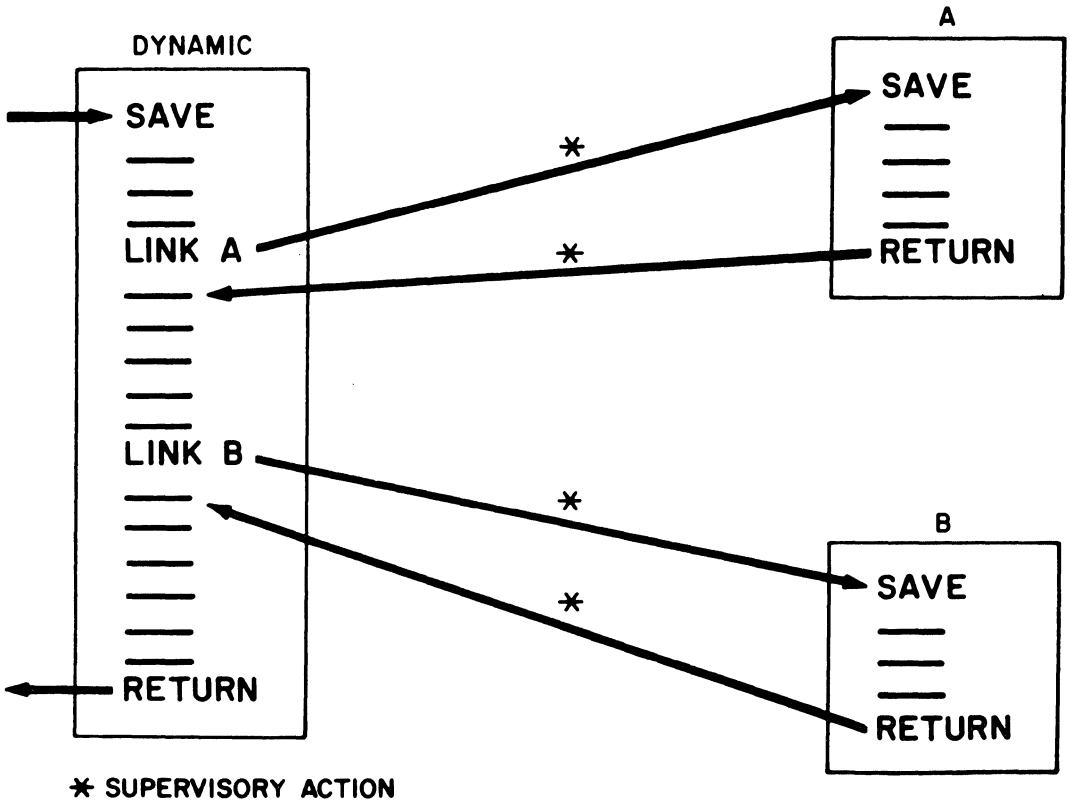
CONTROL BLOCKS – VS2





V.4.6

THE LINK OPERATION



V.4.7

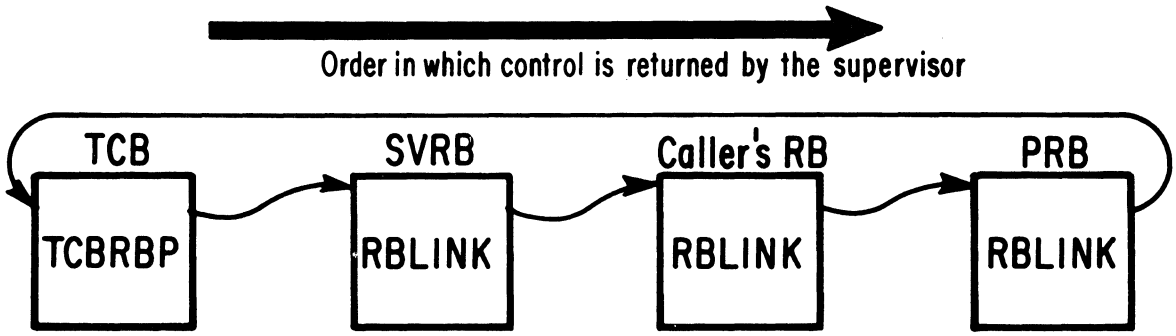
LINK MACRO INSTRUCTION

[symbol] LINK EP=symbol[,DCB=dcb address] [,PARAM=(addresses)] [,VL=1] [,ID=number]

NOTE: There are several additional forms of the macro. Refer to the reference manual.

V.4.8

A REQUEST BLOCK QUEUE



Task Control Block

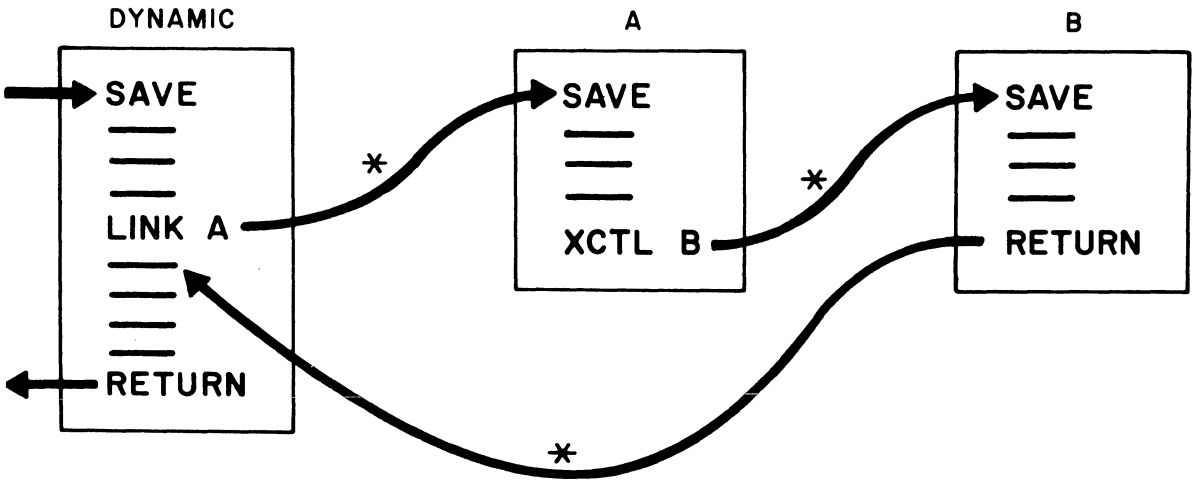
Request block for the currently executing type 2, 3, or 4 SVC routine.

Request block for interrupted routine. (This RB was the current RB before the SVC interruption.)

Request block for the first executed main line program or routine associated with this TCB.

V.4.9

USE OF XCTL MACRO-INSTRUCTION



* SUPERVISORY ACTION

V.4.10

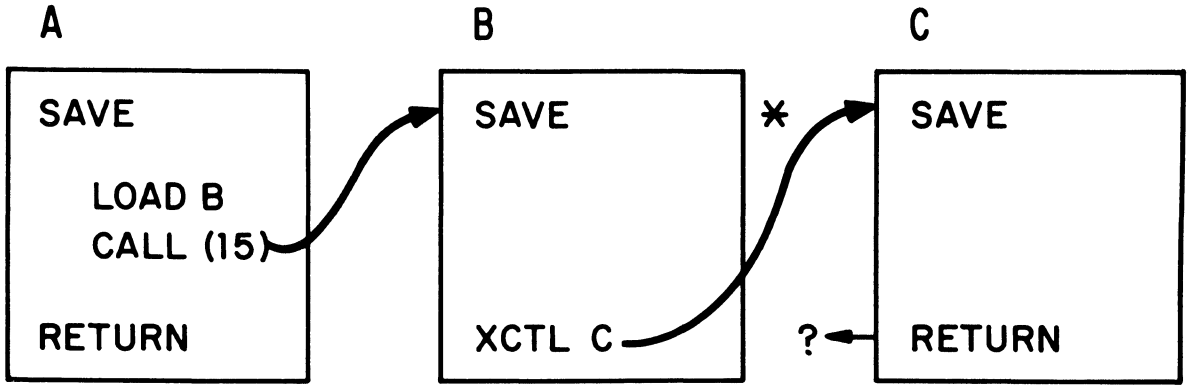
XCTL MACRO INSTRUCTION

[symbol] XCTL [(reg1[,reg2])],EP=symbol[,DCB=dcb address]

NOTE: There are several additional forms of the macro - Refer to the reference manual.

V.4.11

CALL FOLLOWED BY XCTL

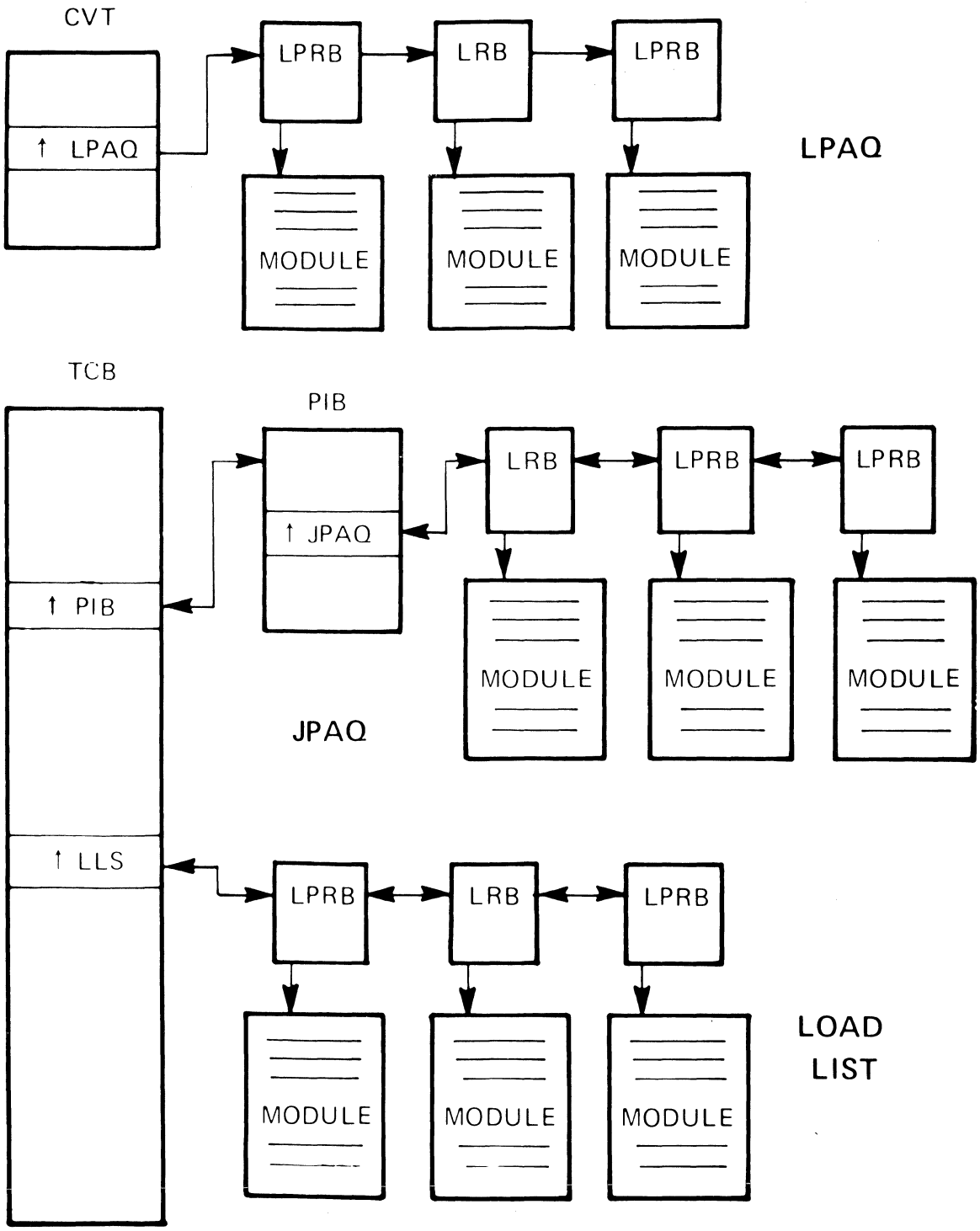


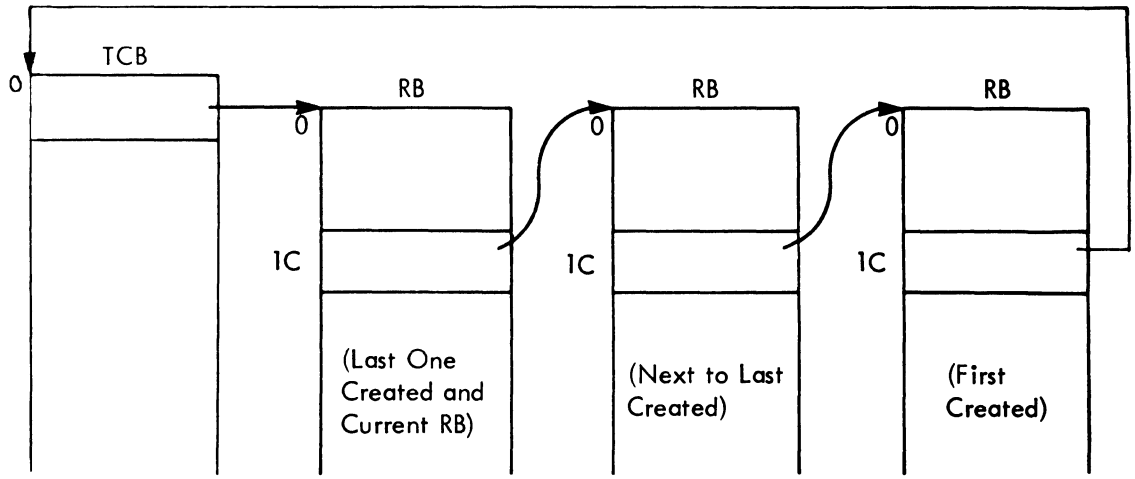
Program A may be destroyed RETURN POINT could be invalid.

* INDICATES SUPERVISION ACTION

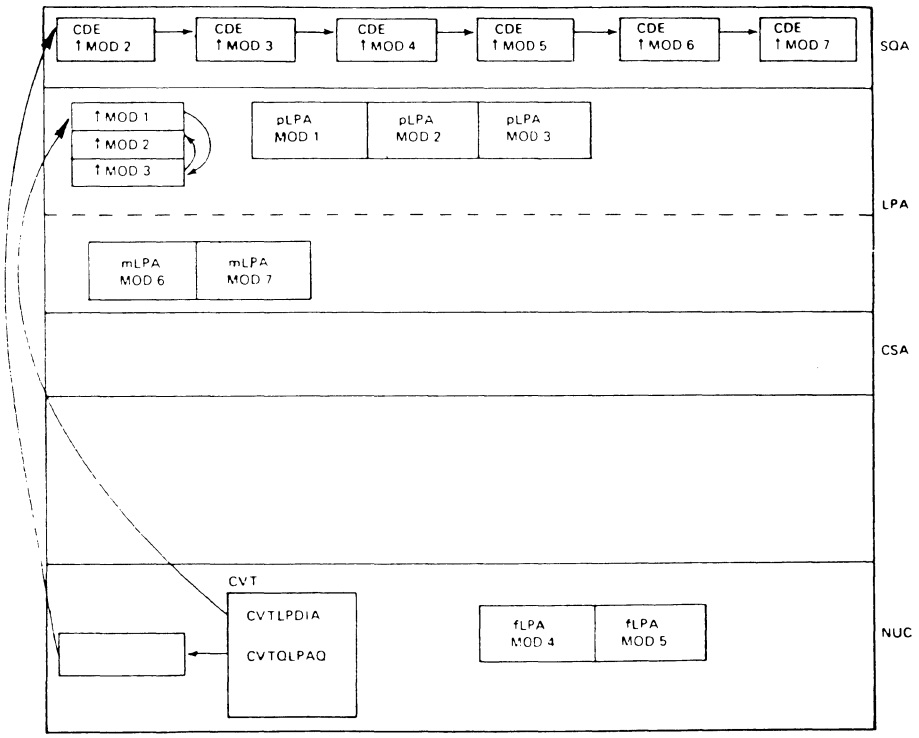
V.4.12

CONTROL BLOCKS – VS1



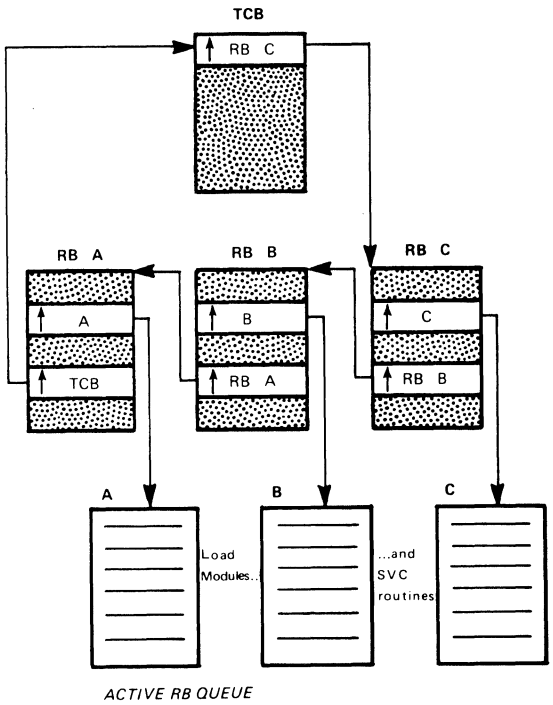

TCB
0
RB
0
1C
(Last One Created and Current RB)
RB
0
1C
(Next to Last Created)
RB
0
1C
(First Created)

V.4.14


CDE ↑MOD 2
CDE ↑MOD 3
CDE ↑MOD 4
CDE ↑MOD 5
CDE ↑MOD 6
CDE ↑MOD 7
SQA
↑MOD 1
↑MOD 2
↑MOD 3
pLPA MOD 1
pLPA MOD 2
pLPA MOD 3
LPA
mLPA MOD 6
mLPA MOD 7
CSA
CVT
CVTLPDIA
CVTQLPAQ
fLPA MOD 4
fLPA MOD 5
NUC

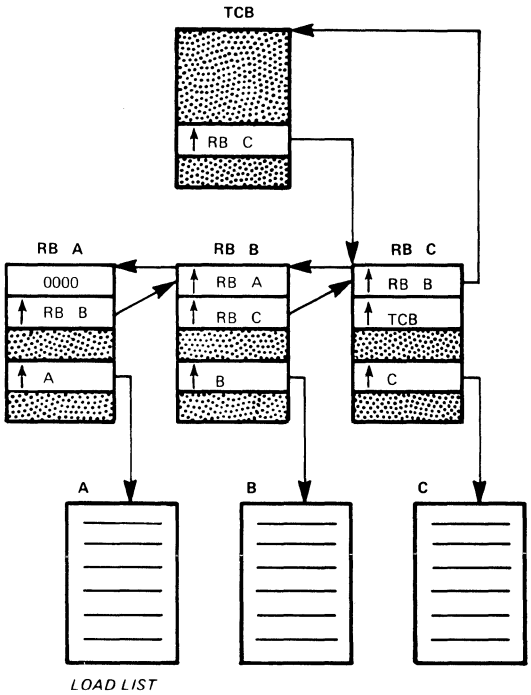
V.4.15

VS1

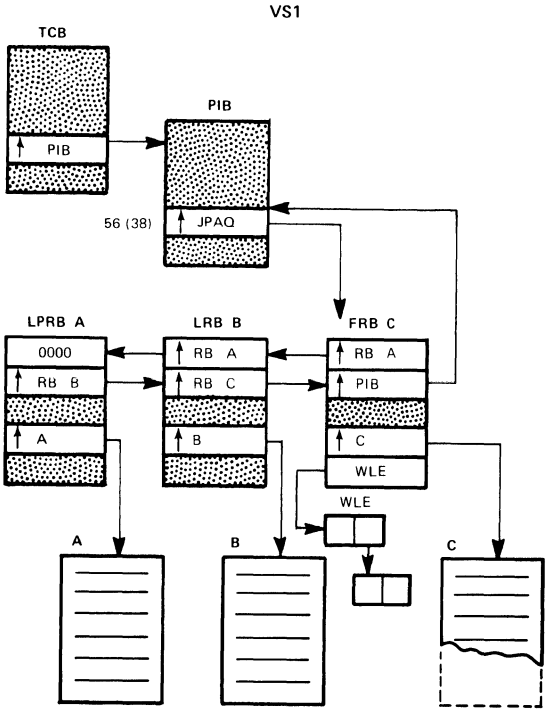


V.4.16

VS1



V.4.17



JOB PACK AREA QUEUE (JPAQ)

TASKS

DEFN:
- A "PROGRAM" CAPABLE OF INDEPENDENT EXECUTION
- COMPETES INDEPENDENTLY FOR CONTROL OF C.P.U.
- IS GIVEN CONTROL BASED ON PRIORITY

TYPES:
- SYSTEM
  - SYSGENED
  - CREATED AT I.P.L. TIME
  - OPERATOR COMMANDS
- PROCESSING PROGRAMS
  - JOB STEP TASKS
  - SUB TASKS

V.5.1

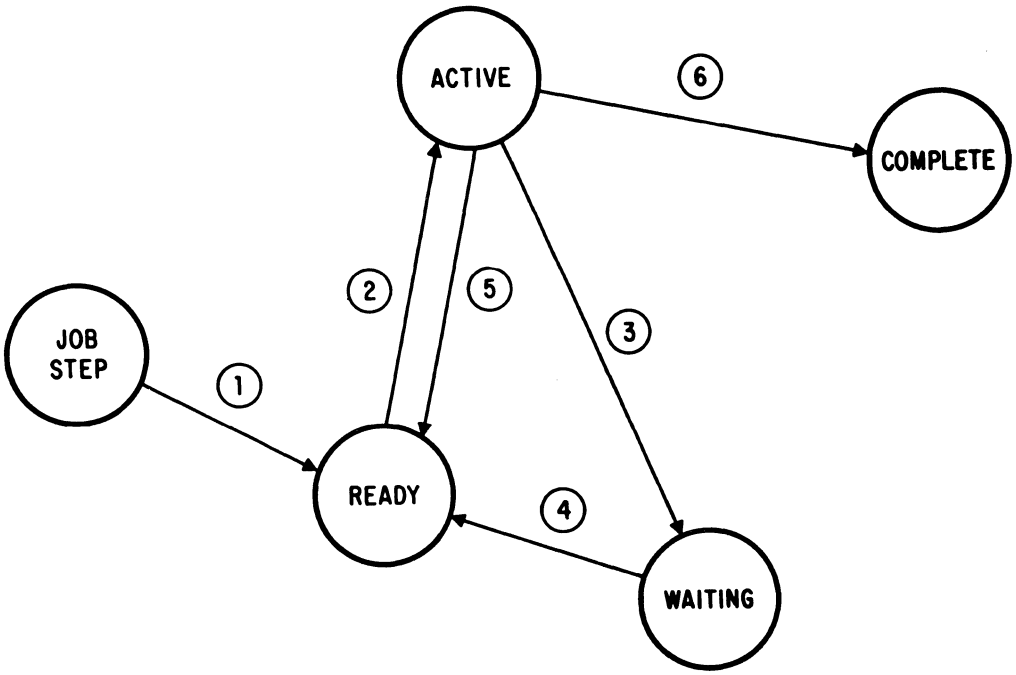
STATES OF A TASK

LEGEND

1. The job step is attached as a task and its task control block is entered into the ready queue.
2. If this ready task has a higher priority than any other ready task, it is dispatched (receives control of the CPU).
3. The task is placed in the wait state to await the completion of some event.
4. The event being waited for is completed so the task is placed in ready state.
5. The active task relinquishes CPU control to a higher priority task that has become ready.
6. A task is completed. Its task control block is deleted from the ready queue and its resources are made available to the system.

V.5.2

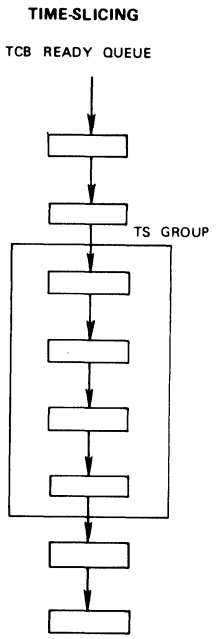
STATES OF A TASK



V.5.3

SYSTEM	PRIORITY	USED FOR	NOW SPECIFIED
VS2-MVS	JOB	JOB SELECTION	JES2 PRIORITY STATEMENT
	ADDRESS SPACE	DISPATCHING THE ADDRESS SPACE	JES2 DEFAULT DPRTY ON EXEC STATEMENT DEFAULTS TO APG
	JOB STEP TASK	DISPATCHING TASKS WITHIN ADDRESS SPACE	SET BY SYSTEM CHANGED BY CHAP
	SUBTASKS	SAME	SET BY ATTACH CHANGED BY CHAP
VS2-SVS	JOB	JOB SELECTION	PRTY ON JOB STATEMENT
	JOB STEP TASK	DISPATCHING TASKS	DPRTY ON EXEC STATEMENT DEFAULT TO JOB PRIORITY CHANGED BY CHAP
	SUBTASKS	SAME	DETERMINED BY ATTACH DEFAULT TO PRIORITY OF ATTACHING TASK CHANGED BY CHAP
VS1	JOB	JOB SELECTION	PRTY ON JOB STATEMENT
	JOB STEP TASK	DISPATCHING TASKS	INITIALLY DETERMINED BY PARTITION PRIORITY CHANGED BY CHAP
	SUBTASKS	SAME	SET BY ATTACH DEFAULTS TO SAME AS CREATING TASK CHANGED BY CHAP

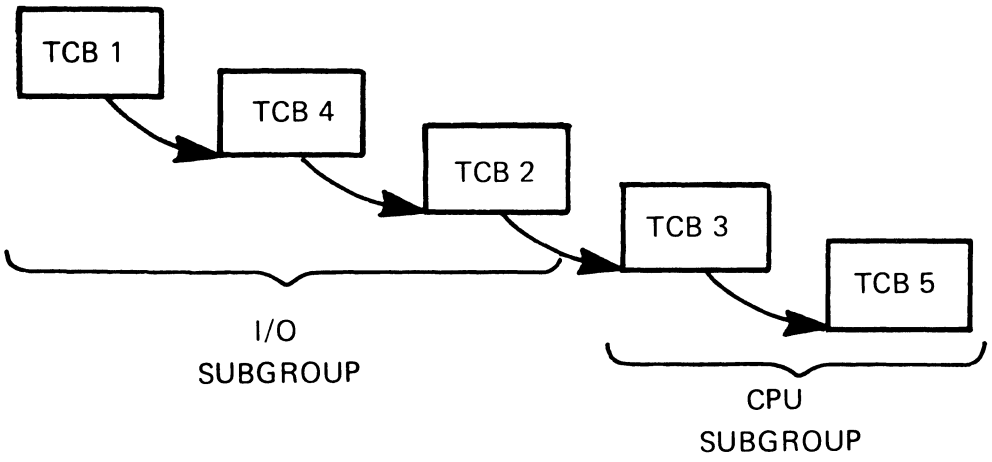
V.5.4



V.5.5

SVS

APG TASKS



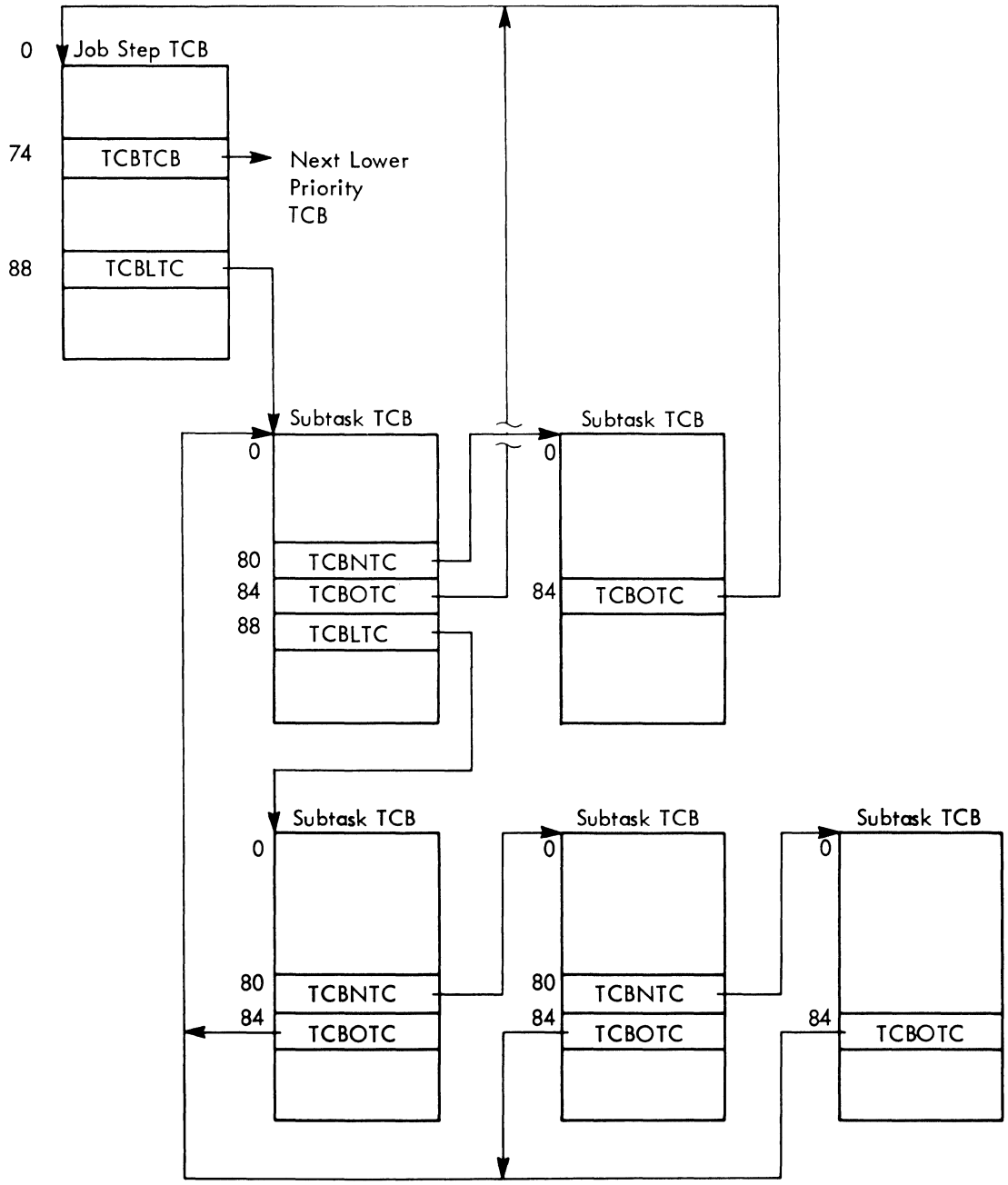
V.5.6

ATTACH MACRO INSTRUCTION

```
[symbol]   ATTACH   EP=symbol[,DCB=dcb address] [,PARAM=(addresses)[,VL=1]]
           [,ECB=ecb address] [,ETXR=exit routine address]
           [,SZERO={YES|NO}] [,TASKLIB=dcb address]
           [,ESTAI=(exit address[,parameter list address])]
           [,PURGE={NONE|HALT|QUIESCE}] [,ASYNCH={YES|NO}]
```

NOTE: There are additional forms of the macro and additional operands. Refer to the reference manual.

TCB RELATIONSHIPS



DETACH MACRO INSTRUCTION

[symbol] DETACH tcb location address[,STAE={YES / *NO*}]

EXTRACT MACRO INSTRUCTION

[symbol] EXTRACT answer address [, {'S' / tcb address}] ,FIELDS=(tcb info)

tcb info:

ALL	PRI	TSO*
GRS	CMC	PSB*
FRS	TIOT	TJID*
AETX	COMM	ASID**

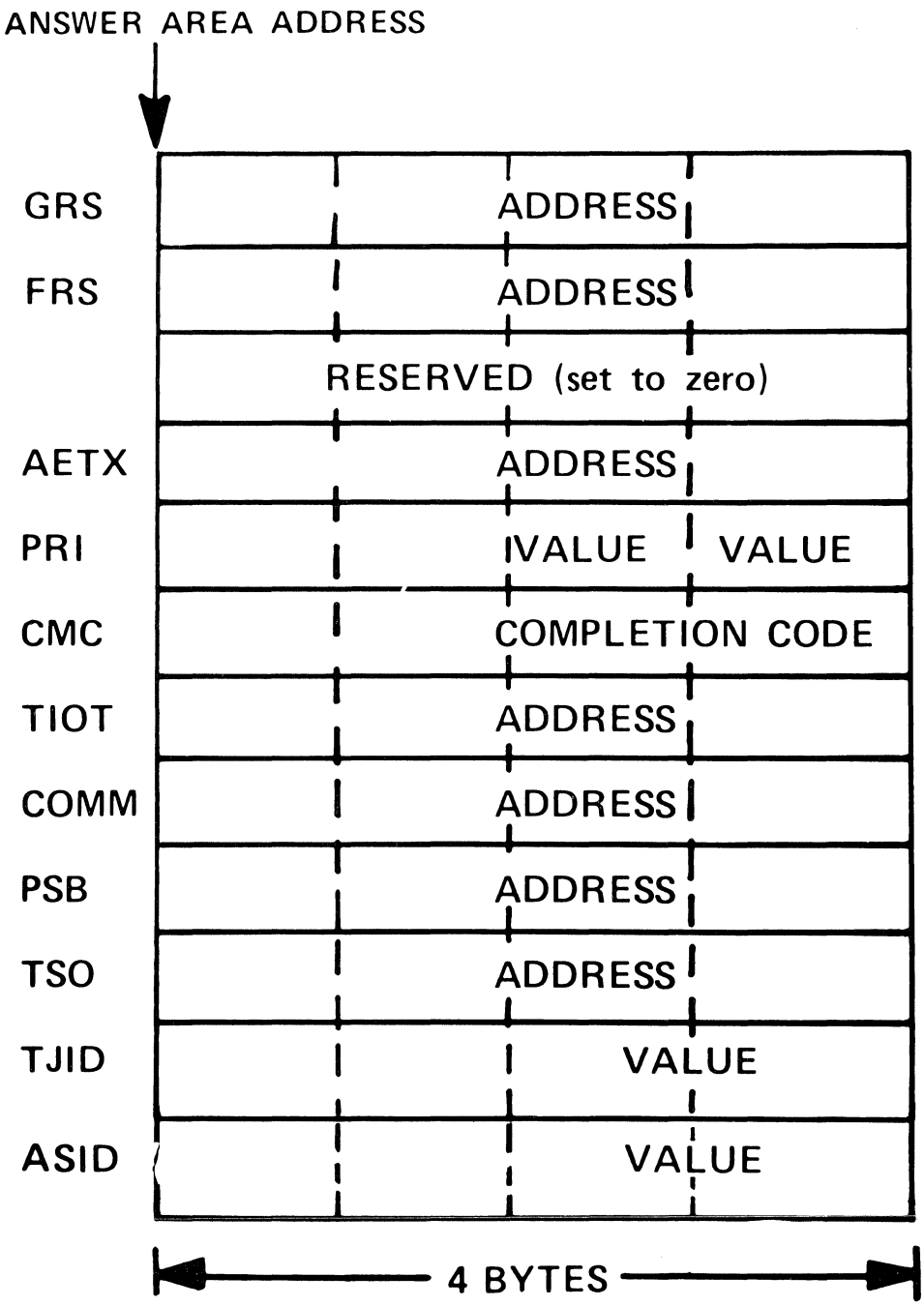
* SVS and MVS ONLY

** MVS ONLY

EXTRACT FIELDS

VALUES	FIELDS TO BE EXTRACTED
ALL	All of the following fields.
GRS	Address of the general register save area.
FRS	Address of the floating point register save area.
	(RESERVED)
AETX	Address of the entry point of the asynchronous termination routine specified by the task that attached the task whose task control block is specified. (This routine is specified by the ETXR operand of the ATTACH macro-instruction.)
PRI	Limit and dispatching priority values. (These values are stored into the third and fourth bytes, respectively, of the list word. The two high-order bytes of this word are set to zero.)
CMC	Task completion code. (If the task has not completed, this field is zero.)
TIOT	Address of the task input/output table (TIOT).
COMM	Address of the command scheduler communications list. The list consists of a pointer to the communications event control block and a pointer to the command input buffer. The high-order bit of the last pointer is set to one to indicate the end of the list.
PSB	Address of the protected storage control block (PSCB), which is extracted from the JSCB.
TSO	Address of the time sharing flags field in the TCB.
TJID	Terminal job identifier (TJID) of the task specified in the TCB location address operand.
ASID	Address space identifier of the address space that TCB is running under.

ANSWER AREA FIELD ORDER, EXTRACT MACRO



CHAP MACRO INSTRUCTION

[symbol] CHAP priority change value [,tcb location address / ,'S']

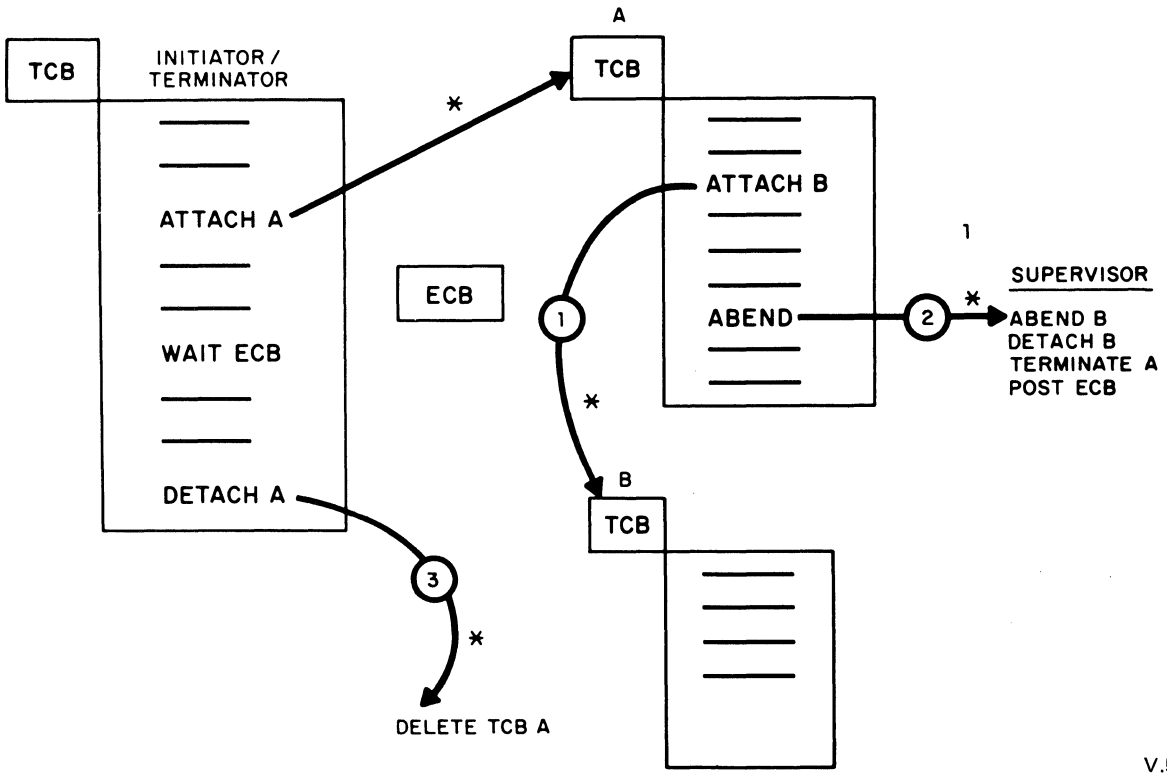
WAIT and POST MACRO INSTRUCTIONS

[symbol] WAIT event number, {ECB=ecb address / ECBLIST=ecb list address}, LONG={NO / YES}*, RELATED=value

[symbol] POST ecb address, completion code, RELATED=value*

*MVS Only

ABNORMAL TERMINATION OF A SUBTASK

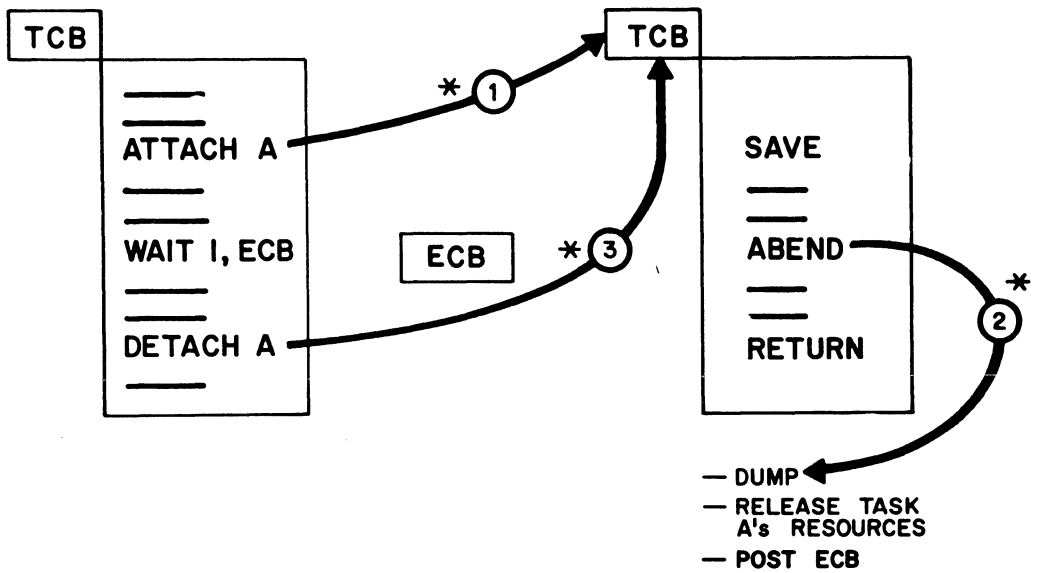


V.5.15

* SUPERVISORY ACTION

ABNORMAL TERMINATION OF A TASK

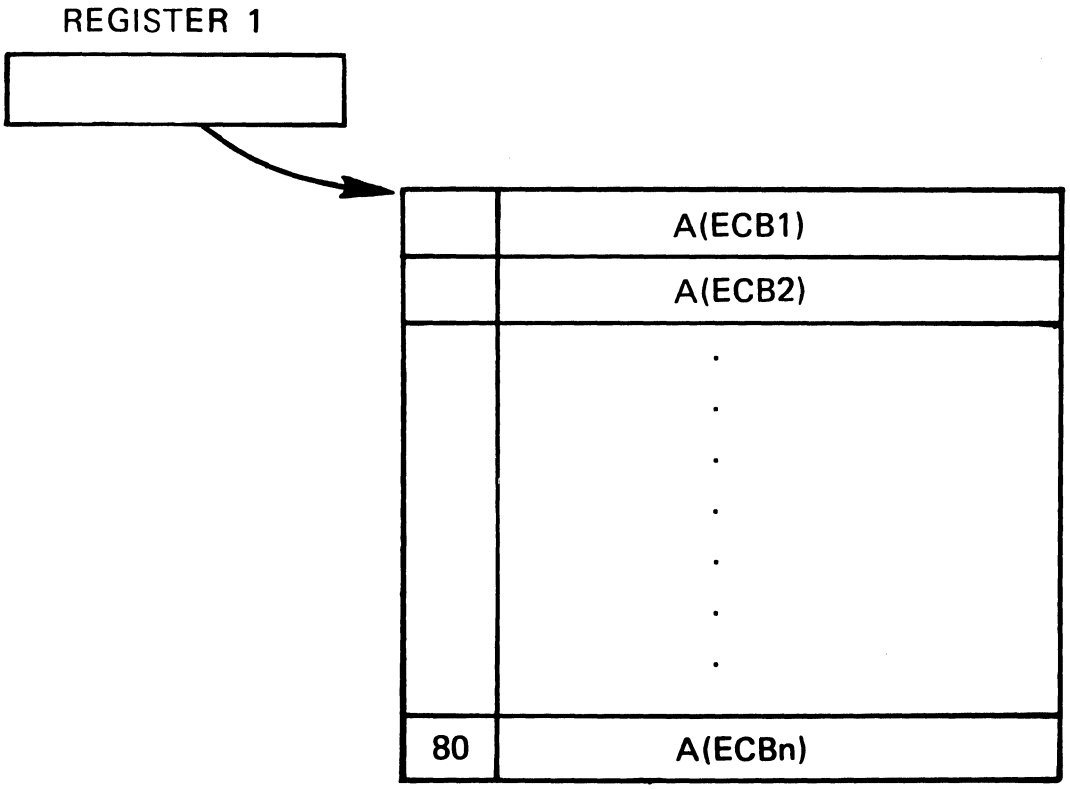
TASK FOR JOB SCHEDULER
INITIATOR/TERMINATOR



V.5.16

* SUPERVISORY ACTION

EVENTS TABLE



V.5.17

EVENTS MACRO INSTRUCTION

```
[symbol]   EVENTS   ENTRIES=n
[symbol]   EVENTS   ENTRIES=DEL,TABLE=table address

[symbol]   EVENTS   TABLE=table address[,WAIT= { NO } ] [,ECB=ecb address]
                                               { YES }
                    [,LAST=last address]
```

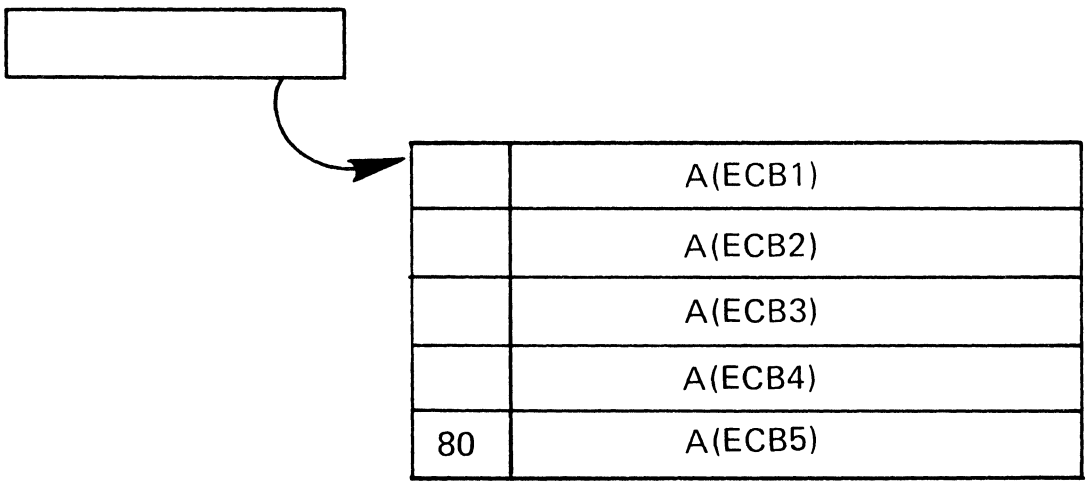
V.5.18

PROCESSING ONE EVENT AT A TIME

```
EVENTS    TABLE=table address,WAIT=YES
```

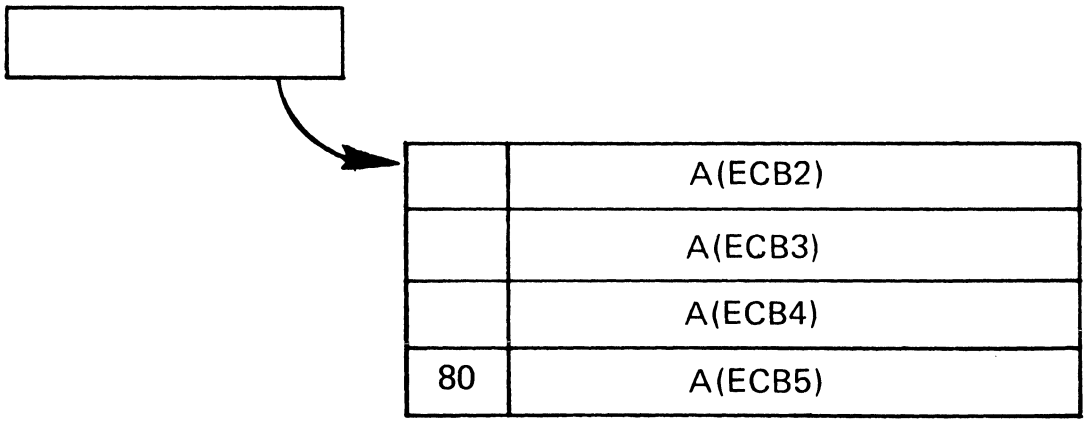
1ST TIME ISSUED:

REGISTER 1



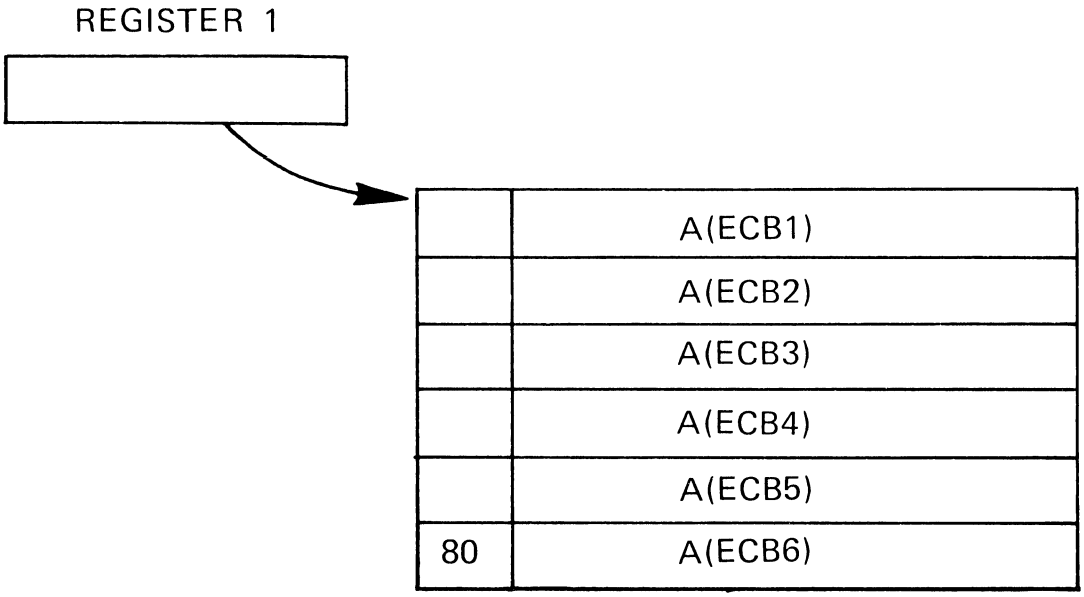
2ND TIME ISSUED:

REGISTER 1



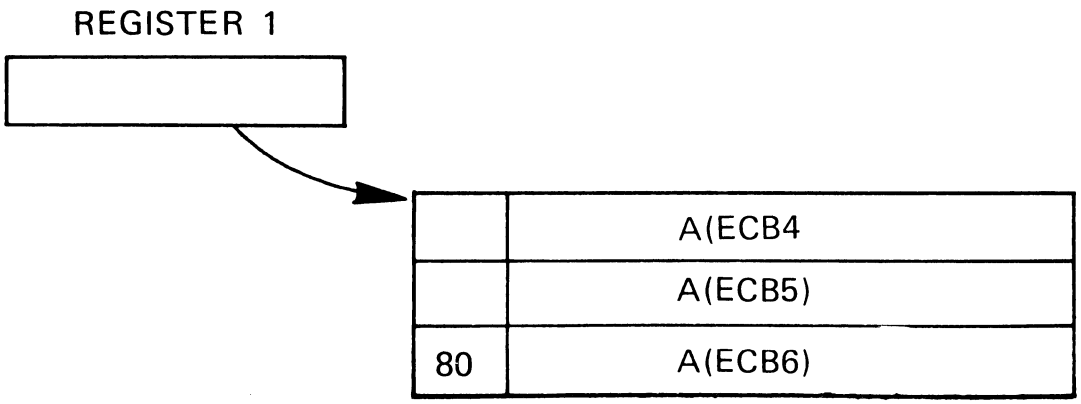
PROCESSING MULTIPLE EVENTS

EVENTS TABLE=table address,WAIT=YES



Assume that 3 events were processed after the above EVENTS was issued and then Register 2 was loaded with the address of the last entry processed (ECB3).

EVENTS TABLE=table address,WAIT=YES,LAST=(2)



PROCESSING MULTIPLE EVENTS - CODE

```
         START
           .
           .
           .
         EVENTS    ENTRIES=20
         ST        1,TABADD
         LA        2,ECBA
         EVENTS    TABLE=TABADD,ECB=(2)
           .
           .
           .
         EVENTS    TABLE=TABADD,WAIT=YES
         LR        3,1          Get Addr of PARM LIST
         B         LOOP2        Go Process ECB
LOOP1    EVENTS    TABLE=TABADD,WAIT=YES,LAST=(3)
         LR        3,1
LOOP2    EQU       *
           .
           .  (Process Completed Events)
           .
         TM        0(3),X'80'   Test for more Events
         BO        LOOP1        IF NONE, GO WAIT
         LA        3, 4(3)      Get next entry
         B         LOOP2        GO Process next entry
           .
           .
           .
EOJ      EVENTS    TABLE=TABADD,ENTRIES=DEL

           .
           .
```

ENQ MACRO INSTRUCTION

```
[symbol]    ENQ    (qname address, rname address, [E/S],
                   [rname length], [SYSTEM / STEP], . . .)
                   [,RET=TEST
                   ,RET=USE
                   ,RET=HAVE
                   ,RET=CHNG]
```


DEQ MACRO INSTRUCTION

```
[symbol]    DEQ    (qname address, rname address,[rname length]
                   , [STEP / SYSTEM], . . .)[,RET=HAVE]
```

DEQ MACRO INSTRUCTION RETURN CODES

CODE	MEANING
0	CONTROL OF THE RESOURCE HAS BEEN RELEASED.
4	CONTROL OF THE RESOURCE HAS BEEN REQUESTED FOR THE TASK, BUT THE TASK HAS NOT BEEN ASSIGNED CONTROL. THE TASK IS NOT REMOVED FROM THE WAIT CONDITION. (THIS RETURN CODE COULD RESULT IF THE DEQ MACRO INSTRUCTION IS ISSUED WITHIN AN EXIT ROUTINE WHICH WAS GIVEN CONTROL BECAUSE OF AN INTERRUPTION.)
8	CONTROL OF THE RESOURCE HAS NOT BEEN REQUESTED BY THE ACTIVE TASK OR CONTROL HAS PREVIOUSLY BEEN RETURNED.

V.6.3

ENQ Return Codes

CODE	MEANING			
	RET=TEST	RET=USE	RET=HAVE	RET=CHNG
0	The resource is immediately available.	Control of the resource has been assigned to the active task.		The status of the resource has been changed to exclusive.
4	The resource is not immediately available.		--------	The status cannot be changed to shared.
8	A previous request for control of the same resource has been made for the same task. Task has control of resource.			The resource has not been queued.
20	A previous request for control of the same resource has been made for the same task. Task does not have control of resource.			Not used.

V.6.4

TIME MACRO INSTRUCTION

```
[symbol]   TIME { DEC                    } ,ZONE= { LT  }
                { BIN                    }        { GMT }
                { TU                     }
                { MIC, storage address   }
                { STCK,storage address   }

                ,ERRET=error routine address
```

V.6.5

STIMER MACRO INSTRUCTION

```
[symbol]   STIMER { REAL [,timer completion exit address] }
                  { TASK [,timer completion exit address] }
                  { WAIT                                  }

                  { ,DINTVL=address    }
                  { ,BINTVL=address    }
                  { ,TUINTVL=address   }
                  { ,TOD=address       }
                  { ,MIC=address       }
```

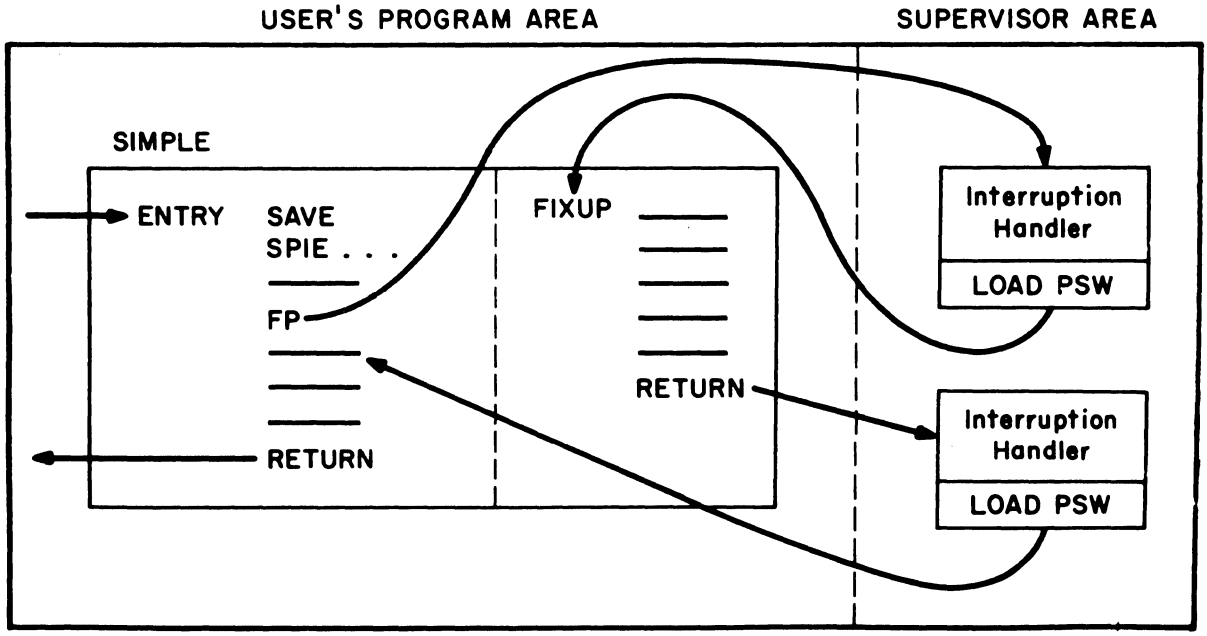
V.6.6

TTIMER MACRO INSTRUCTION

[symbol] TTIMER [CANCEL] [, { TU / MIC,address }]

V.6.7

SET PROGRAM INTERRUPT EXIT



V.6.8

SPIE MACRO INSTRUCTION

[symbol] SPIE exit address,(interrupts)

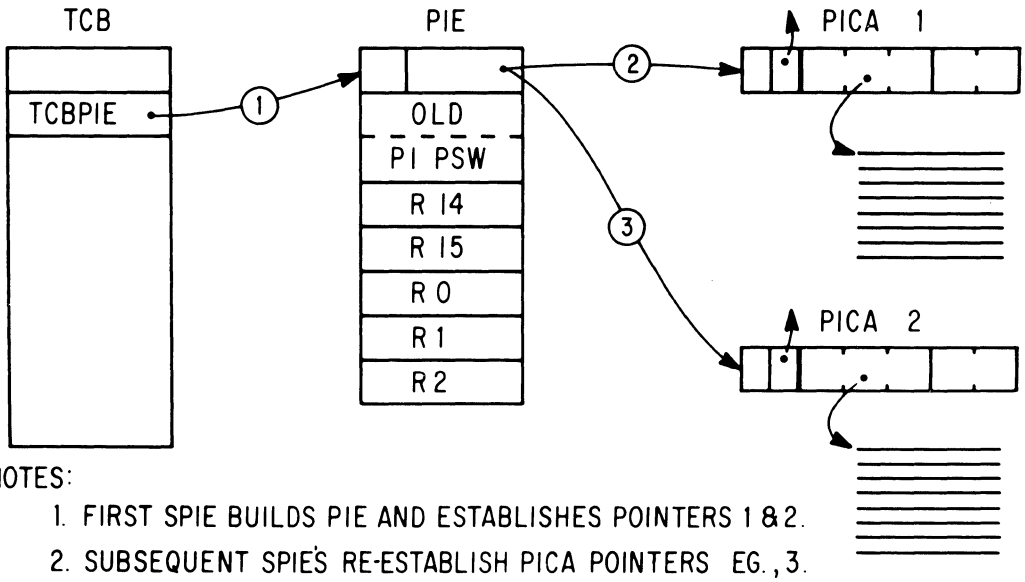
interrupts:

1. Operation
2. Privileged operation
3. Execute
4. Protection
5. Addressing
6. Specification
7. Data
8. Fixed-point overflow (Maskable)
9. Fixed-point divide
10. Decimal overflow (Maskable)
11. Decimal divide
12. Exponent overflow
13. Exponent underflow (Maskable)
14. Significant (Maskable)
15. Floating-point divide

V.6.9

VS1 AND SVS

CONTROL BLOCKS & POINTERS ESTABLISHED BY SPIE SVC ROUTINE -



NOTES:

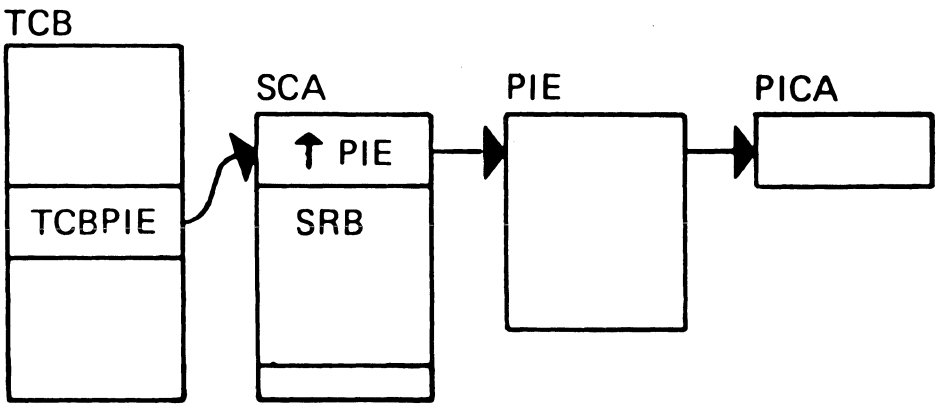
1. FIRST SPIE BUILDS PIE AND ESTABLISHES POINTERS 1 & 2.
2. SUBSEQUENT SPIE'S RE-ESTABLISH PICA POINTERS EG., 3.
3. SPIE WITHOUT OPERANDS PUTS ZERO POINTER IN PIE.

V.6.10a

MVS

CONTROL BLOCKS AND POINTERS ESTABLISHED BY SPIE ROUTINE

SPIE – ROUTINE SCHEDULED BY AN SRB



V.6.10b

PROGRAM INTERRUPTION ELEMENT

BYTES	0	1	2	3
0		PICA address		
4 8	OPSW after interruption			
12	Register 14			
16	Register 15			
20	Register 0			
24	Register 1			
28	Register 2			

V.6.11

STAE EXIT ROUTINE

- STANDARD LINKAGE CONVENTIONS
- CANNOT ISSUE STAE OR ATTACH
- RESIDENT
- FUNCTIONS
  - PERFORM PRETERMINATION PROCESSING
  - DIAGNOSE
  - RETRY
- SCHEDULE RETRY OR CONTINUE WITH ABEND

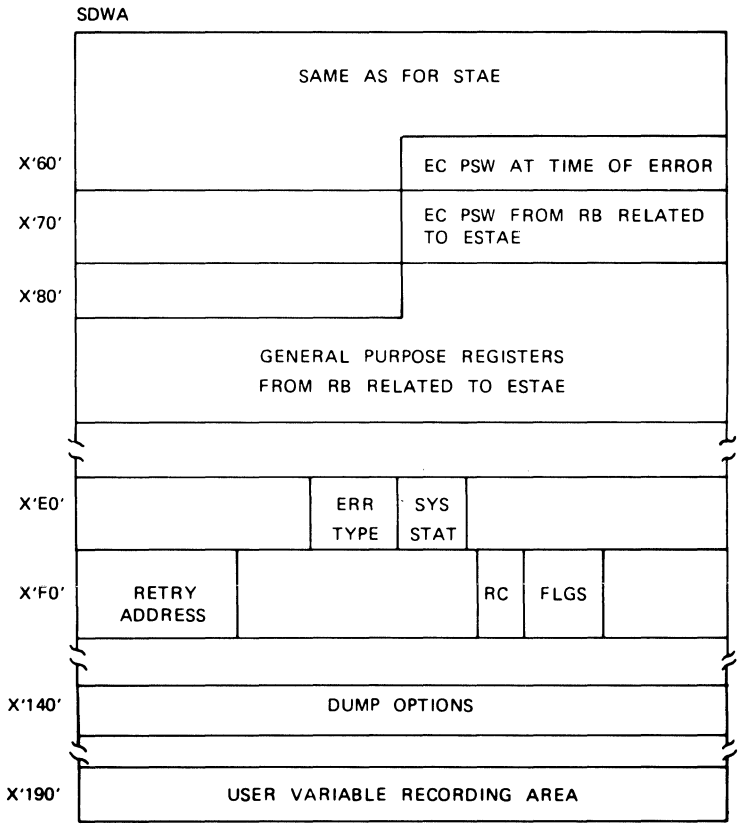
V.6.12

IMPROVED FACILITIES – ESTAE

- INCREASED SIZE FOR SDWA
- SOFTWARE RECORDS ON SYS1.LOGREC
- GTF TRACE
- ESTAE EXIT CAN ISSUE ESTAE
- ESTAE EXITS CAN RECEIVE CONTROL ON 'CANCEL' TYPE ABENDS

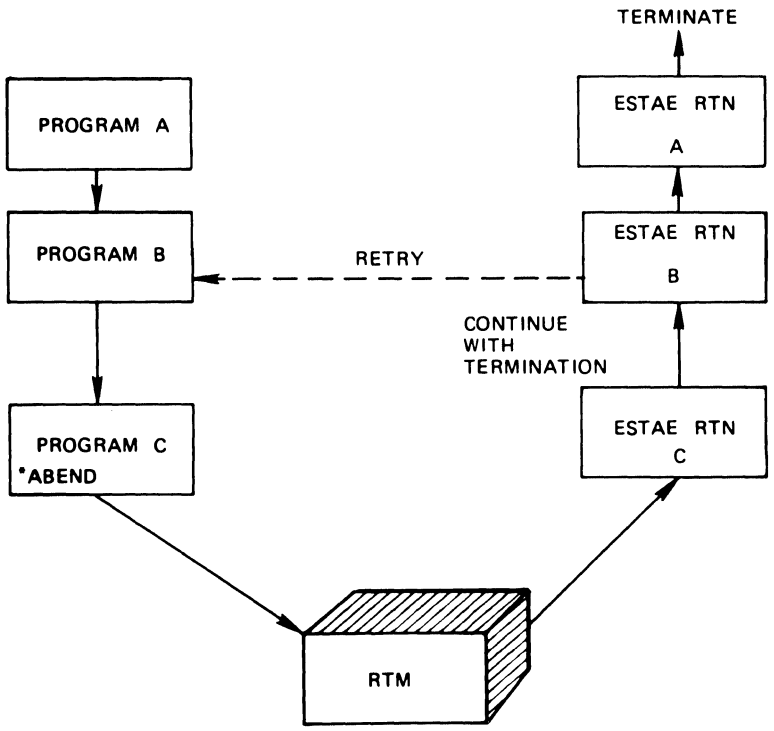
V.6.13

SYSTEM DIAGNOSTIC WORK AREA



V.6.14

PERCOLATION



V.6.15

ESTAE CONSIDERATIONS

- EXIT CAN ISSUE ESTAE or ATTACH
- PERCOLATION
- I/O Options - only for 1st EXIT
- New SDWA for each exit
- For "CANCEL" type of termination
  - TERM=YES must be specified
  - Retry request will be ignored
  - ESTAE macro ISSUED in ESTAE exit will have TERM=YES ignored
  - Only one *STAE* exit will receive control

V.6.16

READER'S COMMENT FORM

OS/VS Multiprogramming Services Student Materials ZR20-4454-1

Please comment on the usefulness and readability of this publication; suggest additions and deletions, and list specific errors and omissions (give page numbers). All comments and suggestions become the property of IBM.

COMMENTS

Reply Necessary

Yes ☐

No ☐

Name _____

Company _____

Address _____

_____ Zip _____

Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

FOLD ON TWO LINES, STAPLE AND MAIL.

YOUR COMMENTS PLEASE. . . . . .

Your comments on the other side of this form will help us improve future editions of this publication. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material.

Please note that requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or the IBM branch office serving your locality.

FOLD FOLD

FIRST CLASS
PERMIT NO. 40
ARMONK, NEW YORK

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY:

IBM Education Center, Building 005
Department 78L, Publications Services
South Road
Poughkeepsie, New York 12602

FOLD FOLD

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

ZR20-4454-1

OS/VS Multiprogramming Services Student Materials Printed in U.S.A. ZR20-4454-1